195 PTAS.
(IVA Incluido)

110

MI COMPUTER

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



P.V.P. Canarias, Ceuta y Melilla 185 Ptas.

9 788485 822836 10110

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen X-Fascículo 110

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-183-6 (tomo 10)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 258602
Impreso en España-Printed in Spain-Febrero 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

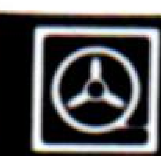
b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Interludio musical

Cada vez más tanto los músicos aficionados como los profesionales utilizan los micros. Demos una mirada a esta popular aplicación

Liz Heaney

Cajas de música
El advenimiento de la síntesis de sonido controlada digitalmente ha popularizado el uso de micros como realizadores de música programables y controladores de instrumentos electrónicos. Los informáticos han comprendido las dificultades con que se encuentran la mayoría de los aspirantes a músicos por ordenador al programar directamente el chip de sonido del micro, y han producido software más sencillo para simplificarles la tarea. Aquí vemos algunos de los paquetes que se comercializan en la actualidad

Los paquetes de música para microordenadores han estado disponibles desde la introducción de los primeros modelos PET y Apple a finales de los años setenta. Estos primeros ejemplares eran algo primarios y de utilidad limitada; pero ese mismo período también fue testigo de la introducción de sintetizadores de música electrónicos y portátiles igualmente elementales diseñados para ser usados en casa y en el escenario. Debido a que la música occidental está estructurada según reglas matemáticas precisas, basándose en el exacto equilibrio en la generación de sonidos en combinación con silencios, muchos músicos apreciaron rápidamente las posibilidades.

Inicialmente, la principal razón que motivó la inclusión de facilidades de sonido en los micros personales fue el proporcionar ruidos adecuados para los juegos. Pero algunos fabricantes, como Commodore, Acorn y Amstrad, comprendieron que existía una demanda de capacidades para hacer música, e incluyeron en sus máquinas chips bastante sofisticados para generar y moldear sonidos. Otros fabricantes, como Sinclair, continuaron ciñéndose a los generadores de tonos de tipo *beep*. Ésta es la razón por la cual hay tan pocos paquetes de música disponibles para el Spectrum, aunque existen algunos accesorios de hardware que añaden sistemas con circuitos para generación de tonos más útiles y proporcionan un software activador adecuado.

Lo más simple para el aspirante a músico por ordenador es programar el chip de sonido del micro. La mayoría de los fabricantes que incluyen sofisticados chips de sonido también suministran instrucciones de BASIC que permiten seleccionar alturas y duraciones de notas y moldear el sonido mediante instrucciones de envoltura. Una notable excepción la constituye Commodore, que no permite acceder a su chip de sonido (presumiblemente el mejor que existe en un micro personal de costo reducido) desde BASIC: este chip se debe programar utilizando una complicada serie de PEEKs y POKEs directamente en los registros internos del chip de sonido.

La mayoría de los chips de sonido disponen de tres voces, lo que permite tocar hasta tres notas al mismo tiempo. Por lo tanto, se pueden producir acordes y piezas a tres voces con facilidad. El control de envoltura de volumen afecta a la calidad de una nota y por lo general se define mediante una larga serie de números, que a su vez definen la altura y el tamaño de paso de cada sección de envoltura. El moldeo de las envolturas de tono permite la adición de efectos más sofisticados, tales como vibrato. Además, suele haber disponibles distintas formas de onda, como triangulares y cuadradas.

El principal problema de la programación de música desde BASIC reside en que es bastante lenta en comparación con el grado de precisión del control de tiempo necesario para producir una música que suene natural. Incluso el sistema de cola de sonido activado por interrupciones de Amstrad sólo consigue atenuar ligeramente este problema. La mejor solución (desde el punto de vista de la programación) es utilizar código máquina; pero, por supuesto, ello hace que las cosas se vuelvan sumamente difíciles excepto para los programadores más meti-

culosos. En consecuencia, numerosas empresas que han detectado el creciente interés por la programación de música han producido interfaces sencillas para personas que deseen tocar música.

Estos paquetes se dividen en dos tipos principales: los que incorporan las facilidades para generación de sonido del ordenador, y los que utilizan al ordenador para controlar un equipo externo de generación de sonido. El ejemplo más evidente de este último tipo es una red de teclados MIDI controlados por un micro. A la primera categoría la podemos subdividir en los paquetes que requieren un hardware adicional, como un teclado, y aquellos que se basan exclusivamente en software.

Existen a la venta muchos paquetes que convierten al ordenador en un instrumento que se puede tocar en tiempo real; es decir, que las pulsaciones del teclado se convierten inmediatamente en los sonidos audibles correspondientes. Los paquetes más económicos utilizan el teclado del ordenador, pero accesorios caros, como el Echo, proporcionan un medio más tradicional para tocar. Tales paquetes suelen incluir métodos para inicializar el chip de sonido para poder alterar la calidad del sonido.

Es aquí donde los microordenadores pueden contribuir enormemente al bagaje del músico, porque es posible escribir una pieza musical paso a paso utilizando la notación estándar o un lenguaje musical especializado de la misma forma en que uno utilizaría un procesador de textos. Esto ofrece la ventaja adicional de permitir que uno escuche y edite la pieza que está componiendo. Además, es posible que el ordenador analice y convierta una pieza tocada en tiempo real al sistema de notación que se esté utilizando. Una vez que el compositor está satisfecho, la pieza ya acabada se puede almacenar para futuras referencias e imprimir en notación estándar mediante una impresora matricial común.

En este caso, no existe ningún paralelismo con el sintetizador especializado, aparte del empleo de compositores o secuenciadores dedicados o el uso de un micro para el mismo fin a través de una MIDI, como la CX5M de Yamaha.

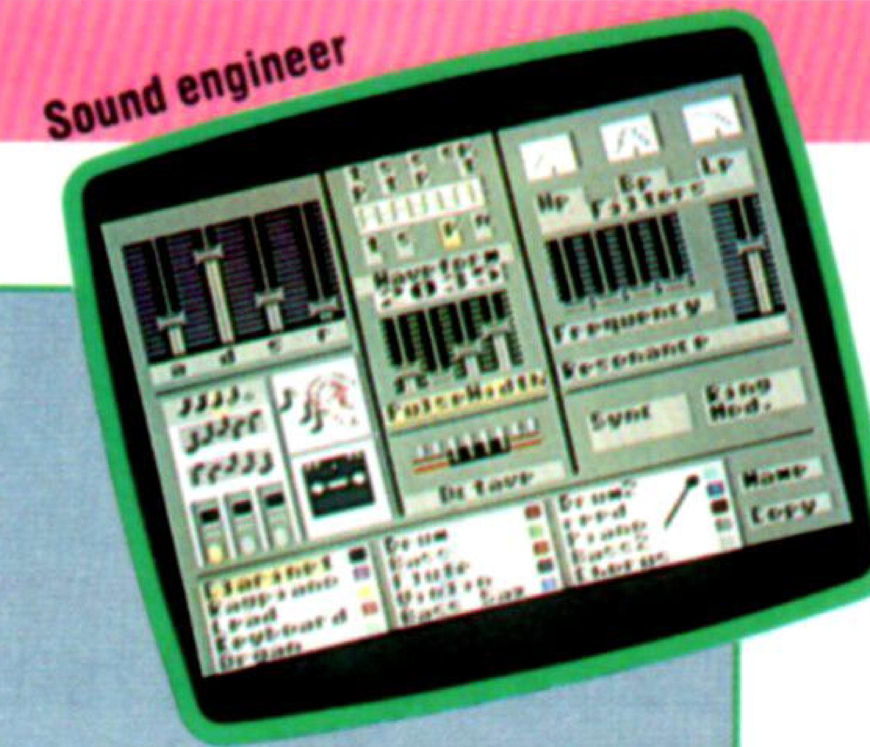
La mayoría de las personas a quienes interesa la programación se sentirán tentadas a programar música y efectos sonoros. El principal atractivo de la mayoría de los paquetes de música tiene una doble vertiente: la programación paso a paso de música permite que aun el más amateur de los usuarios produzca piezas que presenten un cierto grado de dificultad. El auténtico músico también se siente atraído por tales paquetes, ya que permiten la experimentación a través del familiar medio de la notación musical estándar de composición y forma. Sistemas como el CX5M, que combina facilidades de composición con un medio para controlar instrumentos musicales electrónicos externos, maximiza el potencial de este tipo de sistemas.

Para los usuarios de instrumentos MIDI, la inversión adicional, relativamente pequeña, que representan un micro personal y una unidad de disco les proporciona acceso a un método de control, grabación y reproducción simultáneos de hasta 16 instrumentos MIDI. Aunque la mayor parte de los instrumentos MIDI se basan en teclados y máquinas de ritmos, también están adquiriendo notable difusión y aceptación los controladores de guitarras e instrumentos de viento.

Componer música más fácilmente

La programación paso a paso es una facilidad que permite que aun el recién iniciado absoluto escriba sus propias melodías, que el ordenador puede volver a reproducir. Aunque es probable que haya tantas versiones de programación paso a paso como programas para composición musical, las características esenciales son las mismas.

Las notas musicales se entran en el ordenador a través de su propio teclado, y algunas veces las notas van sonando a medida que son introducidas. Las notas se visualizan luego en la pantalla, a menudo en el pentagrama musical estándar de cinco líneas. Una vez que se ha completado la pieza, el usuario puede editar la composición volviendo hacia atrás y alterando una nota, cambiando la clave de tiempo, etc. Este método de composición ofrece una gran ventaja: permite a los usuarios oír lo que están componiendo a medida que van entrando las notas, y pueden disponer de una reproducción instantánea. Además brinda a los aspirantes a compositores la posibilidad de oír sus melodías, las que quizá sean incapaces de ejecutar en un instrumento musical convencional

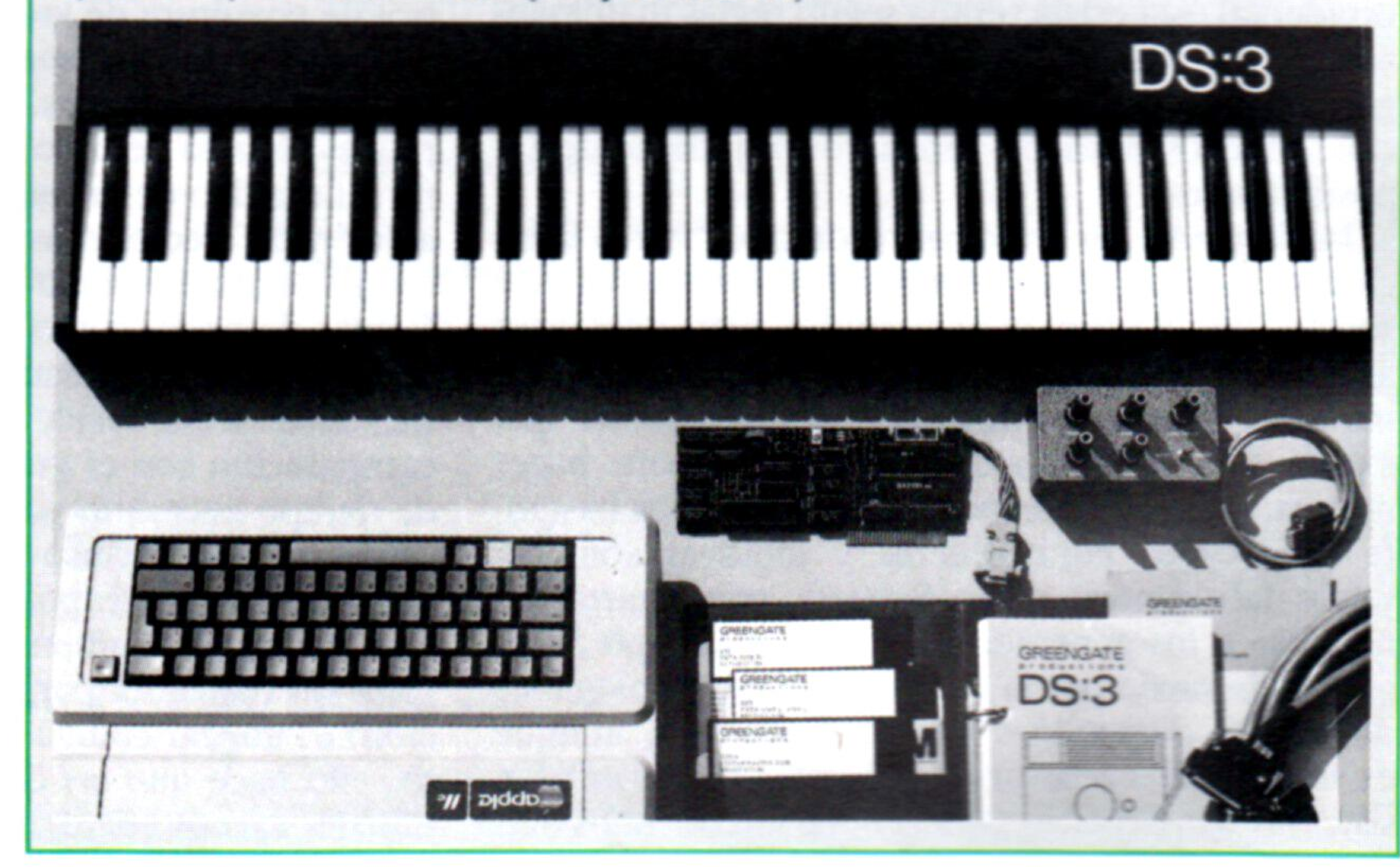


Un programa notable
El *Music studio* de Activision, activado por iconos, permite programar el chip de sonido, además de componer melodías que se pueden reproducir. El *Sound engineer* posee formato tipo sintetizador, ofreciéndole la posibilidad de tocar un instrumento o programar el suyo propio. El programa *Music editor* permite programar melodías paso a paso. Al igual que *Sound engineer*, las facilidades se seleccionan por medio de un cursor "batuta", y las composiciones se escriben en la notación musical estándar

Music editor

Muestreo

El muestreo suele englobarse en la etiqueta general de síntesis de música, si bien en realidad es la conversión y almacenamiento de un sonido (una señal analógica) en una serie de valores digitales. Éstos pueden ser convertidos de nuevo a casi su valor analógico inicial a voluntad, para tocar, componer o secuenciar. Hasta hace poco tal "grabación" digital estaba limitada a sistemas exclusivos muy caros. Ahora se encuentra disponible en el mercado, a un precio asequible, el sistema DS3 para el Apple II, que permite el muestreo polifónico de cuatro notas. Se han desarrollado, a precios también muy convenientes, algunos sistemas monofónicos para el Commodore 64 (Autographics Microsound 64) y para el Spectrum (Ricoll Sound Sampler y Datel DSS)



Sonido sintetizado

Al igual que los ordenadores, los primeros sintetizadores eran máquinas enormes. Se basaban en circuitos de osciladores para producir uno o dos sonidos derivados de formas de onda de impulso simple, triangulares o aserradas, individualmente o en combinación. Otro sistema de circuitos moldeaba la envoltura de volumen de tales sonidos y proporcionaba diferentes tipos de filtros de señal para alterar el carácter de los sonidos, al eliminar de la señal frecuencias seleccionadas.

Esto dotaba a los sintetizadores de un cierto grado de versatilidad, pero su gran tamaño significaba que sólo podían ser instalados en estudios profesionales. Asimismo, su utilidad como instrumento musical era un tanto limitada, porque los sonidos que producían tendían a ser "chillones", tenues y descoloridos. Fue en estas máquinas donde nació la idea de sintetizadores como instrumentos de teclado tipo piano, en los que cada nota se pudiera marcar como "encendida" o "apagada" mediante la operación de un único interruptor debajo de la tecla correspondiente.

Estos primeros instrumentos portátiles eran monofónicos y operaban de acuerdo a los mismos principios que los voluminosos modelos de estudio, pero economizaban en peso, costo y tamaño, al emplear circuitos integrados; asimismo, seguían siendo enteramente analógicos.

Las técnicas digitales se introdujeron a comienzos de los años ochenta para estabilizar las frecuencias generadas por los osciladores analógicos, dado que éstos tendían a "desviarse" de la melodía. A ello le siguió enseguida la adición de circuitos de memoria para almacenar los ajustes del potenciómetro y los interruptores, permitiendo recuperar e implementar los parámetros para un sonido determinado en cualquier momento con sólo tocar una tecla. Esto es esencial para cambios rápidos de sonido durante una actuación en vivo.

Inicialmente, debido al elevado costo de los chips de memoria, sólo se podían memorizar entre 4 y 12 sonidos. Los sintetizadores modernos tienen capacidad para recordar centenares de disposiciones distintas. Asimismo, la producción masiva de chips de sonido ha significado una drástica reducción de precios.

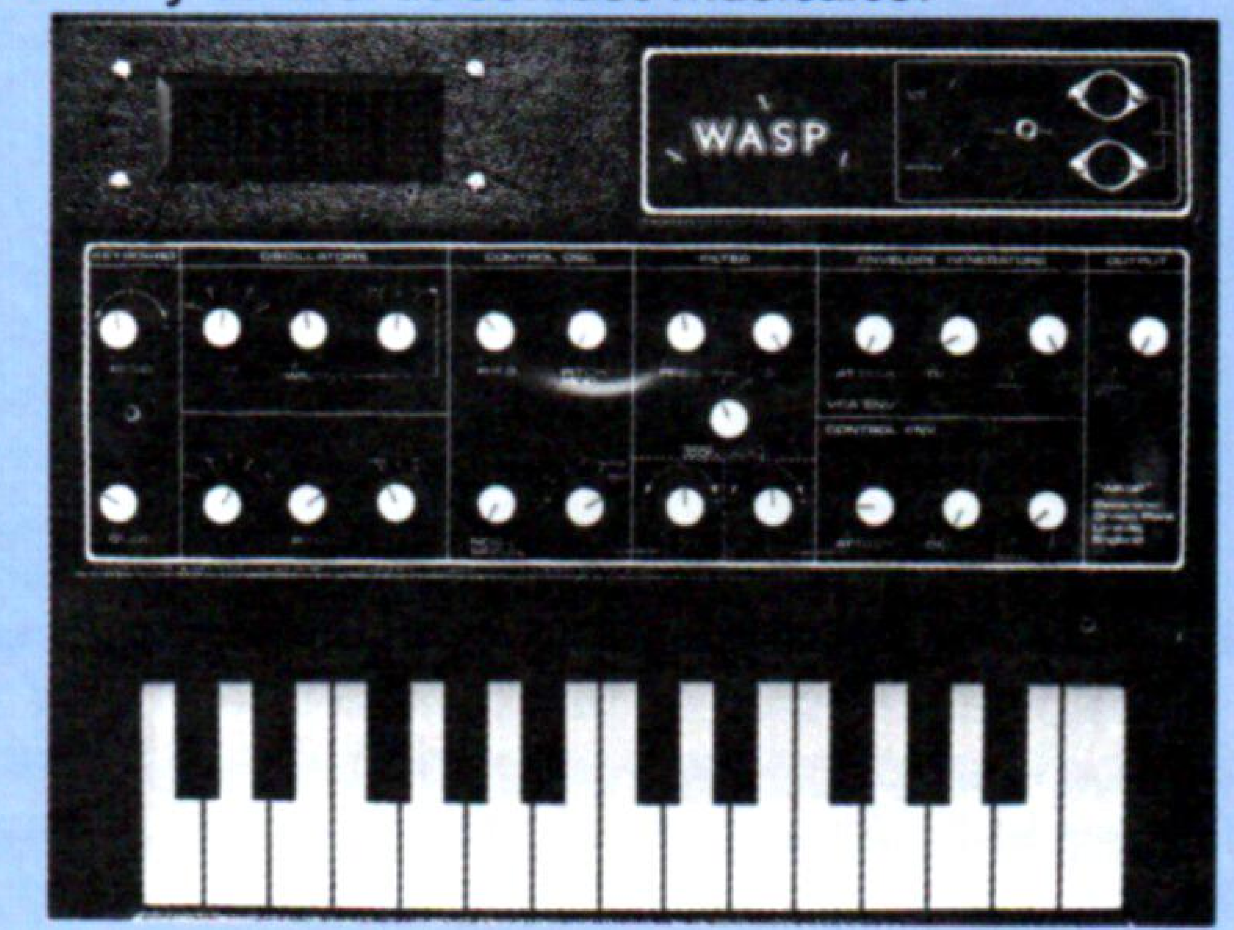
La digitalización de los valores de parámetros para cada circuito modelador de sonido permitió que muchos fabricantes recortaran aún más los costos eliminando los potenciómetros, mandos e interruptores convencionales en favor de una cantidad mínima de interruptores de membrana. Ello significó que se pudiera "llamar" un determinado parámetro y visualizar su valor actual como un número LED con teclas "+" y "−" para modificar el valor. De modo que hacia fines de 1982 los principales fabricantes tuvieron que acordar una interface y sistema de transferencia de datos estándar para que los sintetizadores se pudieran conectar entre sí y entre ordenadores con fines de control.

En agosto de 1983 se estableció un sistema estándar que se dio a conocer como MIDI. Éste en realidad es un estándar de software en virtud del cual las transmisiones MIDI poseen ciertos significados estandarizados. El vínculo entre dos instrumentos especificados por la MIDI no es más que un sistema en serie asíncrono que ya se venía implementando con total éxito desde hacía varios años en sistemas de ordenador, permitiendo que los mismos se comunicaran con periféricos. Uno de los primeros modelos que apareció equipado con MIDI fue, asimismo, el primer sintetizador portátil totalmente digital, el Yamaha DX7, un sistema de ordenador especializado en la generación y control de sonidos musicales.




Minisistema
El MiniMoog, que vemos en la fotografía, fue el primero de una serie de sintetizadores analógicos portátiles económicos basados en la tecnología del circuito integrado





Un sintetizador a su alcance
El desarrollo de sintetizadores de música y ordenadores personales ofrece algunos paralelismos. El sintetizador Wasp, que vemos aquí, puso el costo de los sintetizadores electrónicos al alcance de la mayoría de las personas, al prescindir del oneroso teclado mecánico y reemplazarlo por una económica membrana plástica. El Wasp apareció casi al mismo tiempo que los históricos ordenadores personales ZX80 y ZX81 de Sinclair, que también incorporaban teclados de membrana

Las velocidades de transmisión que se especifican en la MIDI son tales que un micro personal estándar puede actuar como receptor, emisor o procesador intermedio de bytes MIDI. Por último, se ha sellado el vínculo entre la tecnología del ordenador y el equipo de sintetizador profesional. Existen a la venta numerosos paquetes que permiten conectar un micro con un sistema MIDI para permitirle actuar como la unidad de control maestra en un sofisticado sistema de composición, reproducción y grabación

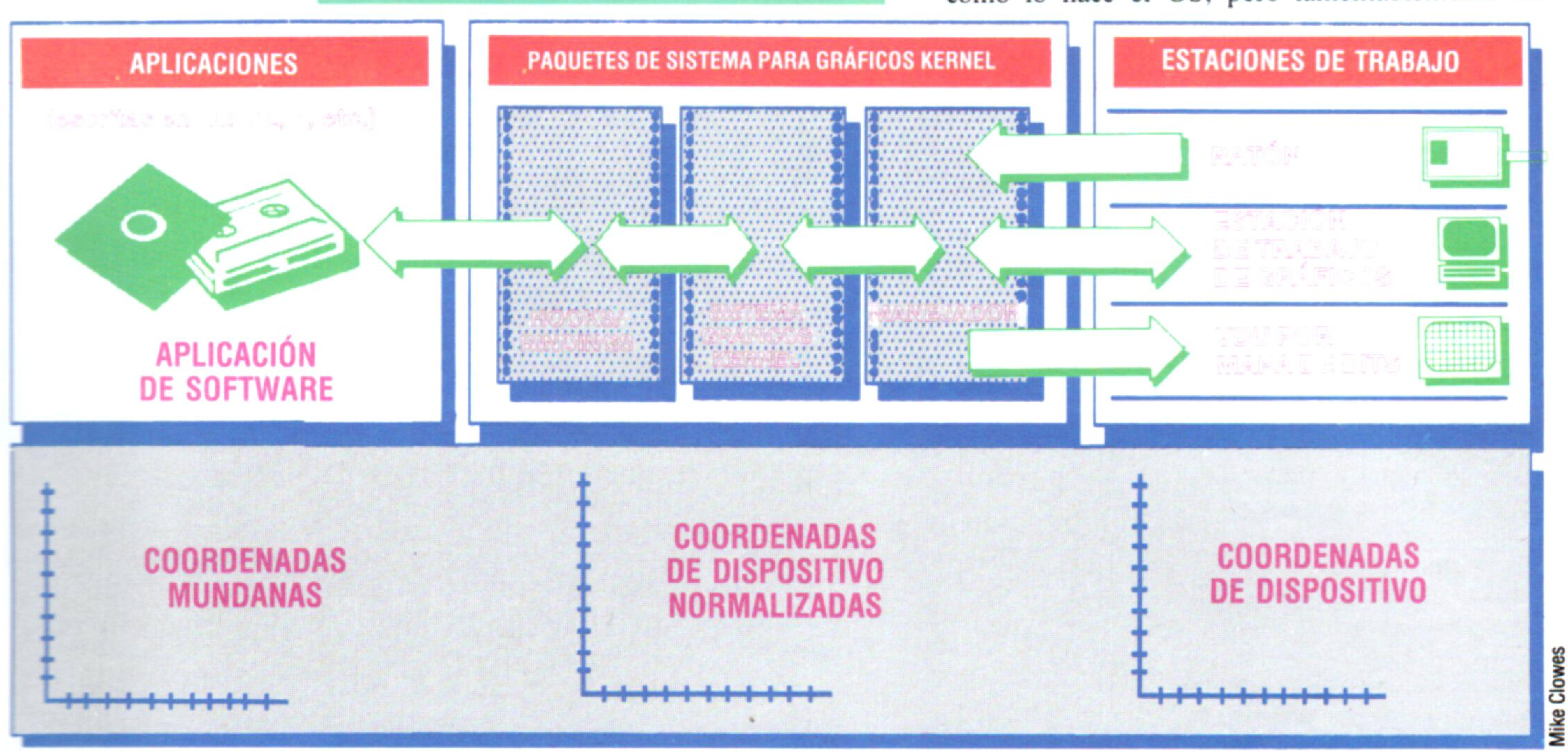


Yamaha DX7

Marcus Wilson-Smith

Saliendo de su concha

El sistema de gráficos «kernel» es una forma de abordar el problema de la portabilidad de las implementaciones de entornos WIMP



Mike Clowes

Bien coordinado
El GKS y otros sistemas de interface para gráficos de bajo nivel obtienen la portabilidad traduciendo las coordenadas del mundo real entradas por el usuario a coordenadas de dispositivo aceptables para el hardware. Mediante este enfoque, sólo necesitan depender de la máquina los manejadores de dispositivo

La implementación de un entorno tipo WIMP para un ordenador es una operación costosa, con un mercado limitado para el producto final. Además, no resuelve el problema clave de hacer que las aplicaciones gráficas sean totalmente portables. La portabilidad implica que el sistema operativo normal tenga un "caparazón" a su alrededor, controlando las ventanas de la pantalla de gráficos, el dispositivo apuntador y también la ejecución por separado de los programas seleccionados. Cuando termina cada operación, el control se vuelve a pasar al caparazón WIMP supervisor en vez de al sistema operativo normal.

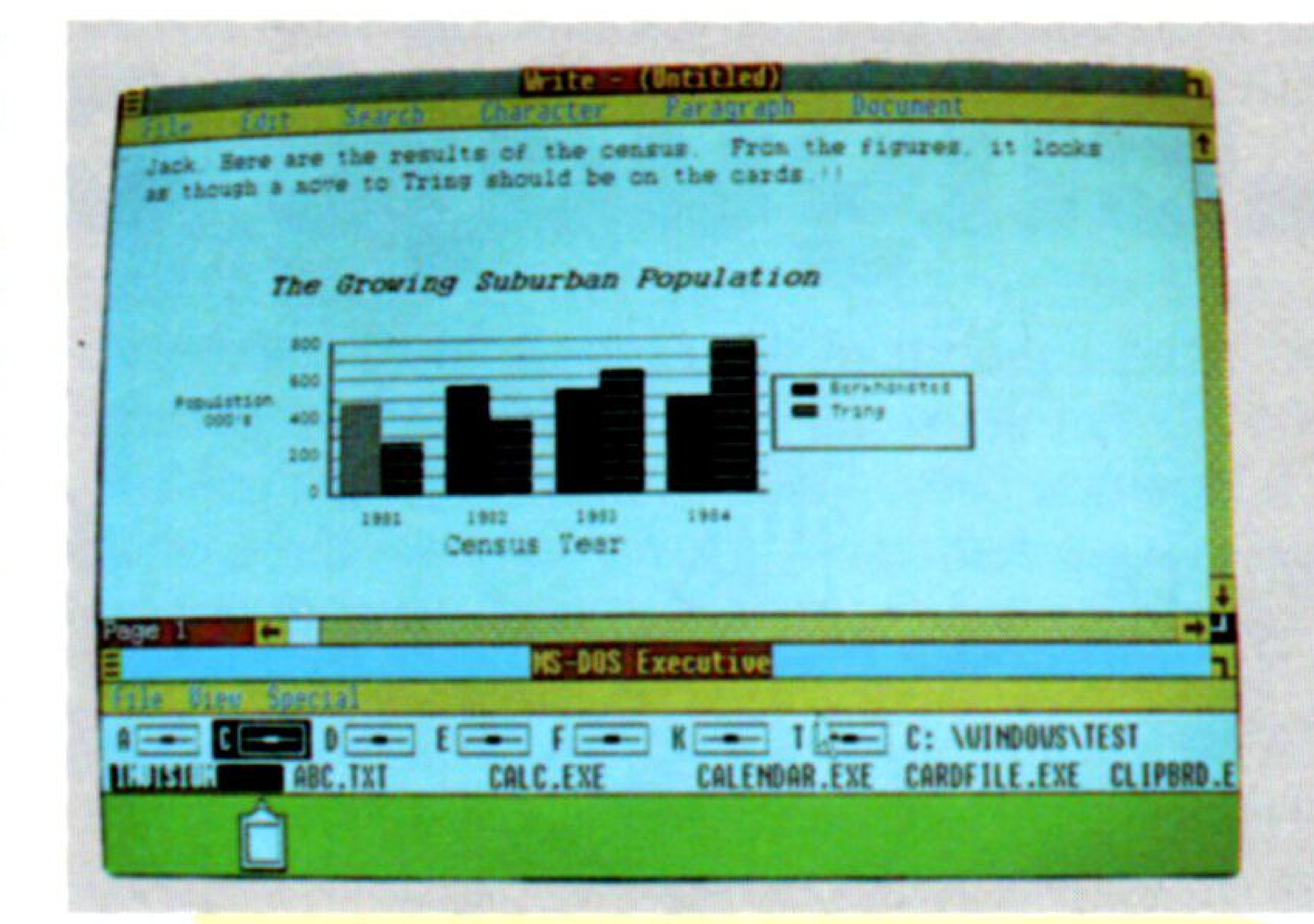
En este sentido, el programa controlador en realidad es un sistema operativo, pero aun así puede llamar al sistema operativo nativo para núcleos de tareas de mantenimiento tales como tratamiento de archivos, etc. La diversidad del hardware de gráficos disponible no armoniza con un entorno coherente de esta naturaleza, pero en la actualidad hay varios esquemas para proporcionar caparazones de gráficos para una amplia gama de máquinas que están luchando por hacerse un sitio en el mercado.

En 1977 se publicaron las primeras especificaciones para un *sistema kernel para gráficos* (GKS) que permitiría la programación de gráficos de una forma independiente de la máquina. Ahora el GKS está disponible a través de varias fuentes, pero su complejidad y elevado precio limitan su utilización a los usuarios de universidades y de aplicaciones de CAD/CAM serias. Digital Research fue la primera empresa especializada en microordenadores que siguió con su caparazón GSX (*graphics system extension*), que se basaba en GKS pero no era compatible con el mismo.

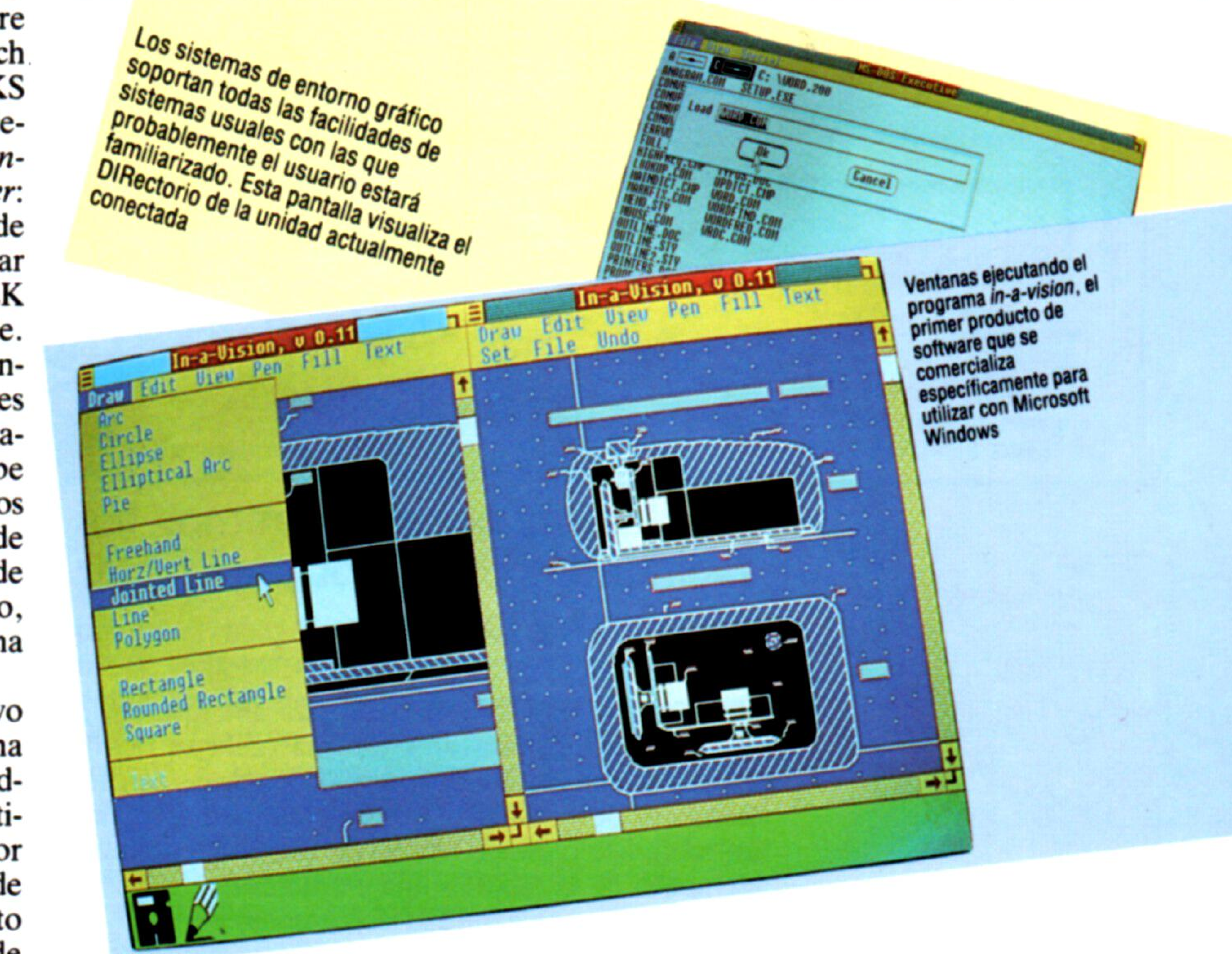
El GSX es una ampliación al sistema operativo (el CP/M-86 de la propia DR), que se construye a medida para cada máquina. Una vez implementado, las aplicaciones pueden llamar a rutinas GSX sin preocuparse ni tener que saber nada acerca de las verdaderas capacidades de hardware, software y firmware utilizadas para llevar a cabo cualquier operación. El GSX descansa sobre el hardware tal como lo hace el OS, pero lamentablemente no posee ninguna interface para el usuario; ésta ha de ser realizada (¡con suma dificultad!) por el programador de aplicaciones.

Una empresa de software británica proporciona una biblioteca de rutinas para acceder a las facilidades GSX sin los problemas de escala, mapa de coordenadas y desbordamiento de capacidad, que constituyen las pesadillas del programador de GSX. Prospero Software suministra tanto compiladores de ISO PASCAL como de FORTRAN 77, y su biblioteca Prospect se puede enlazar con programas escritos en estos lenguajes y ejecutar sin ninguna alteración en cualquier máquina que posea una implementación GSX. Ello proporcionará activadores para una amplia variedad de dispositivos tales como entradas por ratón, tablilla y teclado, así como dispositivos de salida para impresora, plotter y VDU. La biblioteca Prospect accede a las facilidades GSX incluyendo trazado de puntos y líneas, sombreado y, si el dispositivo lo permite, escala y rotación de textos, anchura de línea y color.

Incluso con una interface para gráficos independiente de la máquina y una biblioteca de programas relativamente amable como el Prospect de Prospero, la tarea de implementar gráficos exige un enorme esfuerzo de programación. Por cuanto concierne al usuario medio, la única interacción que se



El color puede suponer una gran diferencia en el tratamiento de imágenes y el resaltado de mensajes en la pantalla, aun cuando la aplicación sea para fines monocromáticos, como ésta, en la que una gráfica desarrollada con un programa se transfiere a un documento de tratamiento de textos



Los sistemas de entorno gráfico soportan todas las facilidades de sistemas usuales con las que probablemente el usuario estará familiarizado. Esta pantalla visualiza el DIRectorio de la unidad actualmente conectada

Ventanas ejecutando el programa *In-a-vision*, el primer producto de software que se comercializa específicamente para utilizar con Microsoft Windows

puede dar es con una aplicación, tanto si utiliza gráficos como si no, y tanto si éstos son activados o no por GKS, GSX, Prospect, FORTRAN, PASCAL, etc.

Para poder implementar un sistema WIMP completo, la totalidad del caparazón debe proporcionar acceso a todos los recursos de un ordenador, no tan sólo a los gráficos y no sólo a través de aplicaciones individuales o programas de sistema. Tanto Digital Research como Microsoft están siguiendo este camino con GEM y MS-Windows, respectivamente. Aparte de IBM, otros fabricantes de hardware que están utilizando GEM y MS-Windows en sus máquinas son Apricot y RML (en el Nimbus).

El *sistema kernel de gráficos*, en el cual se basa el GEM, surgió del deseo de desarrollar un software para gráficos que fuera portable. Digital Research implementó el GSX como un subconjunto del GKS no estándar pero asequible y desde entonces ha llevado la idea adelante con el *administrador del entorno gráfico* (*graphics environment manager*: GEM). El GEM es, en consecuencia, producto de dos corrientes principales de desarrollo: el estándar GKS y la metáfora del escritorio SMALLTALK popularizada por el Lisa y el Macintosh de Apple.

Al igual que el CP/M proporciona un OS estándar a través del cual el software de aplicaciones puede controlar una configuración de hardware variable, un *kernel* para gráficos como el GEM debe envolver con un caparazón blando bien definido los dispositivos físicos de un sistema WIMP. Uno de los conceptos fundamentales es el de la estación de trabajo. Ésta puede ser casi cualquier dispositivo, desde un ratón o trazador/plotter hasta un sistema de E/S completo como un terminal de gráficos.

Por cuanto concierne al GEM, a cada dispositivo del sistema se accede exactamente de la misma forma, y las características y limitaciones de hardware quedan ocultas en el "activador de dispositivo" para esa estación de trabajo determinada. Por consiguiente, a un dispositivo de salida se le puede ordenar que dibuje una determinada forma, tanto si se trata de un plotter como de una pantalla de rayos catódicos. Debido a que la construcción física de cada dispositivo hardware afecta las capacidades y la resolución de los gráficos, se necesita un sistema de coordenadas independiente del dispositivo para aislar las aplicaciones gráficas del hardware. El método del GEM se basa en GKS, que define tres conjuntos de coordenadas.

Cada dispositivo, por supuesto, debe trabajar en función de su propio juego de *coordenadas de dispositivo*. Mediante analogía, una pantalla en color, por ejemplo, tendrá distintas escalas cartesianas en los ejes X e Y de acuerdo a la resolución de la VDU. Las coordenadas y atributos verdaderos del dispositivo son los únicos parámetros de un sistema que opera bajo GEM que varían en función del hardware disponible.

Las coordenadas del dispositivo se transforman a partir de un conjunto de coordenadas "normalizadas" que utiliza el GKS para todas las operaciones de gráficos. Este segundo conjunto de *coordenadas de dispositivo normalizadas* (NDCs: *normalised device co-ordinates*) a su vez se asocia con las coordenadas utilizadas por todo software de aplicaciones en el mundo real. Éstas se conocen como *coordenadas mundanas* (WCs: *world co-ordinates*) y el programador de aplicaciones las puede definir en cualquier escala conveniente. El nivel NDC del sistema GKS de tres hileras está normalizado a una escala de números reales entre cero y uno, de modo que la única limitación en cuanto a resolución es la aritmética de números reales.

En los grandes ordenadores centrales, la potencia y la velocidad disponibles palian la mayor parte de los problemas, pero con los microordenadores las cosas no son tan fáciles. La aritmética de números reales puede consumir mucho tiempo en comparación con la de números enteros, de modo que, en este sentido, el GEM se distancia del sistema GKS. La NDC del GEM responde a una escala entre cero y 32 767, permitiendo almacenar y manipular cada valor en una palabra máquina (16 bits), y permitiendo la minimización del consumo de tiempo introducido por las transformaciones.

El GEM paga un precio pequeño por la velocidad, debido en gran parte a las coordenadas normalizadas de enteros que utiliza, y ello se ve compensado con creces mediante el ahorro que se obtiene en el tiempo de desarrollo del software. Por ejemplo, una vez escrita la biblioteca inicial de rutinas gráficas para el GEM, cualquier aplicación podría incorporarlas y hacer un uso cabal de la capacidad incorporada para manejar los iconos, ventanas, etcétera.

Visualización por ventana
MS-Windows es un sistema de entorno gráfico producido por Microsoft y confeccionado especialmente para alcanzar la compatibilidad con otras aplicaciones y sistemas Microsoft. Las pantallas que vemos aquí muestran el sistema en pleno funcionamiento. Tanto GEM como MS-Windows hacen un uso cabal del color, a diferencia del Apple Macintosh, aunque esto conlleva una disminución de la resolución de pantalla

Lenguaje comercial

—Y ASÍ LLEGAMOS AL FINAL DE NUESTRO CURSO SOBRE TRATAMIENTO DE ARCHIVOS EN COBOL...

BUNCE

Finalmente consideraremos las facilidades del COBOL para tratar archivos y tablas

Entre una tabla y una matriz existe una diferencia: el acceso a los elementos de una tabla se realiza mediante una clave KEY (parte de los datos de cada elemento de la tabla) y no directamente mediante un subíndice o índice. Además, normalmente cada elemento de la tabla es una estructura de registro, conteniendo una cantidad de campos.

Esta clase de estructura se puede manipular en algunos otros lenguajes (el PASCAL, p. ej.) mediante el uso de matrices de registros. El COBOL posee una facilidad para proporcionar tablas, que también se puede utilizar para proporcionar matrices simples. La característica esencial es la idea de que cualquier dato, excepto en nivel 01 o 77, se puede definir mediante repetición añadiendo la cláusula OCCURS nn TIMES (se produce nn veces), donde nn puede ser cualquier constante entera positiva, y TIMES se puede omitir. Por ejemplo, la siguiente define una matriz simple de 20 enteros:

```
01 matriz-simple.
   02 elemento-matriz PIC 9(5) OCCURS 20 TIMES.
```

Los elementos de la matriz son referenciados del modo normal en sentencias, como en elemento-matriz (5) o elemento-matriz (número-1), donde número-1 es un dato entero positivo. Tenga presente, no obstante, que la matriz en su totalidad también tiene un nombre y que, por lo tanto, las operaciones tales como MOVE se pueden llevar a cabo sobre la totalidad de la matriz al mismo tiempo.

Se puede definir una matriz bidimensional (o más grande) dividiendo aún más el campo repetido en un componente que se repita a sí mismo:

```
01 matriz-bidimensional.
   02 fila-matriz OCCURS 20 TIMES.
      03 elemento-matriz OCCURS 30 TIMES.
```

Los elementos de esta matriz son referenciados como elemento-matriz (3,4), por ejemplo, o como elemento-matriz (número-1,número-2); pero se puede referenciar cada fila, como en fila-matriz(6), o la matriz en su totalidad.

Se pueden definir estructuras más complejas (que hagan esta facilidad para tablas en vez de para meras matrices) combinando elementos repetidos de varios tipos con la facilidad del COBOL para subdividir datos. Por ejemplo, la siguiente definición es para una tabla de precios de diversos artículos del stock de una zapatería, que vienen en 20 números distintos:

```
01 tablas-stock.
   02 descripciones-números OCCURS 20 TIMES.
      03 número-inglés PIC 9V9.
      03 número-métrico PIC 99V99.
   02 cantidad-de-artículos-en-existencia PIC 999.
   02 artículos-en-existencia.
      03 artículo-del-stock OCCURS 500 TIMES,
                            ASCENDING KEY IS
                            número-stock
         04 número-stock PIC X(6).
         04 descripción-stock PIC X(20).
         04 precios-stock PIC 999V99 OCCURS
            20 TIMES.
         04 indicador-stock PIC X.
         88 en-existencia VALUE "S".
```

Observe cómo toda la información relevante se mantiene junta en una gran tabla. Cuando se repite un elemento del grupo, como artículo-del-stock en esta definición, también se repiten todos sus subcampos, de modo que podemos referirnos a número-stock (6) y a precios-stock (100,3). Asimismo, el nombre de la condición de nivel 88 puede llevar subíndice, de modo que podemos utilizar IF en-existencia (120).... por ejemplo. La cláusula ASCENDING (o DESCENDING) KEY es opcional; sólo se



utiliza cuando se ha de mantener la tabla ordenada y, entonces, sólo si se ha de utilizar el verbo SEARCH ALL. Es responsabilidad del programador mantener los elementos ordenados, dado que el COBOL no maneja esto de forma automática.

Los datos usados para indexar la tabla pueden ser cualquier elemento numérico, siempre que el valor que contenga sea un entero positivo. Pero por razones de conveniencia y eficacia, el COBOL proporciona un tipo de datos especial, INDEX, y con cada cláusula OCCURS podemos añadir INDEXED BY nombre-índice. Esto define un elemento-dato numérico que sólo se puede utilizar para almacenar valores enteros positivos, y para indexar la matriz con la cual se define. Se puede definir más de un índice para una tabla y se puede declarar cualquier elemento numérico como USAGE INDEX, en cuyo caso puede ser usado para indexar una matriz en uso o en aritmética de índices.

Los elementos de datos índice no se pueden utilizar en las sentencias aritméticas comunes, sino que poseen su propio verbo aritmético, SET:

```
SET nombre-índice TO valor-numérico.
SET nombre-índice UP BY valor-numérico.
SET nombre-índice DOWN BY valor-numérico.
```

donde el valor numérico puede ser una constante o cualquier elemento numérico ordinario.

Una de las principales funciones que es necesario llevar a cabo con una tabla (al contrario que en una matriz) es la búsqueda. El hecho es que, puesto que el acceso se determina por medio de un valor clave, es necesario buscar a través de la tabla con el fin de encontrar la entrada con una clave determinada. El COBOL proporciona esta facilidad directamente mediante el verbo SEARCH (buscar). Éste posee dos formas:

```
SEARCH nombre-tabla VARYING nombre-índice (o
   elemento-numérico)
   AT END sentencia-imperativa
   WHEN condición-1 sentencia-imperativa.
```

donde la cláusula VARYING y la cláusula AT END son opcionales con un número cualquiera de cláusulas WHEN. Esto provoca una búsqueda lineal de la tabla (observando a cada elemento de uno en uno, empezando por el primero), permitiendo que usted especifique la acción a emprender cuando se satisfaga una determinada condición o cuando la búsqueda haya concluido. Una sentencia imperativa es toda sentencia que hace que se tome una acción directa, sin ninguna alternativa. De modo que, por ejemplo, un IF no es una sentencia imperativa, pero un MOVE sí lo es.

La forma alternativa del verbo SEARCH es:

```
SEARCH ALL nombre-tabla
   AT END sentencia-imperativa
   WHEN condición-1 sentencia imperativa.
```

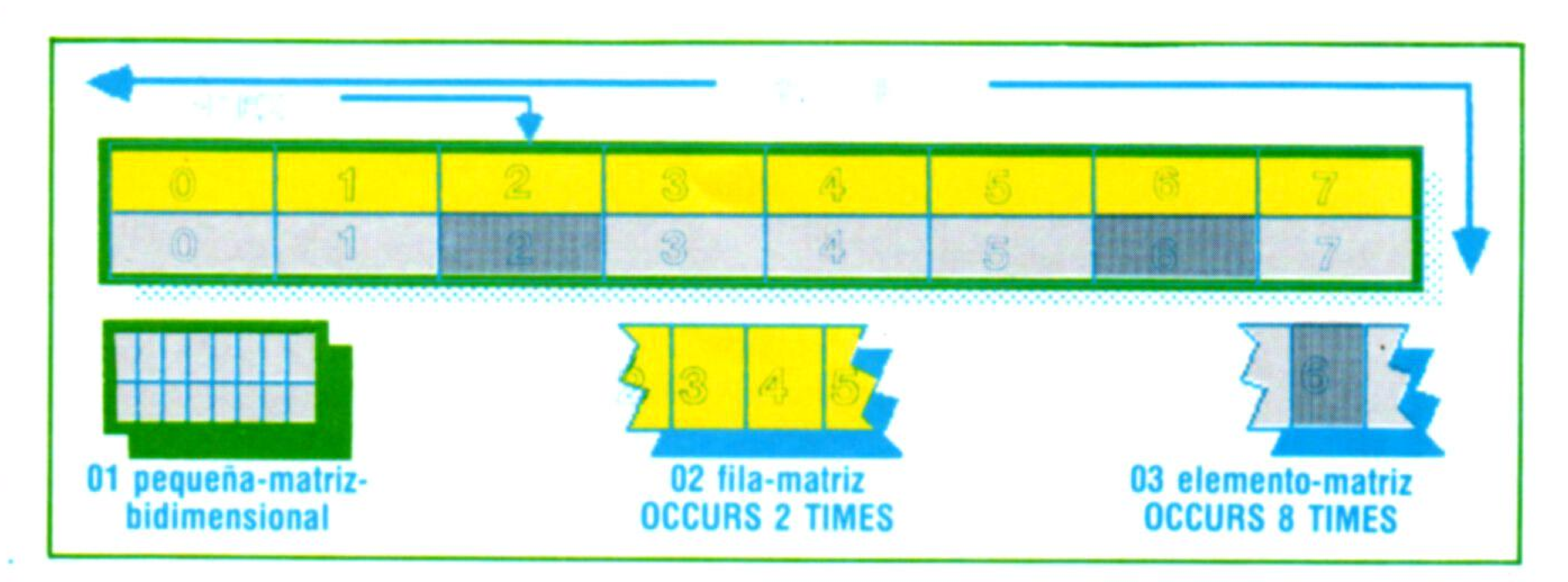
En este caso, sólo puede haber una cláusula WHEN y la comprobación debe tener la forma:

```
elemento-dato=valor
```

o un número de ellas conectadas mediante AND. La diferencia entre las dos formas del verbo es que en este caso se realiza una búsqueda binaria en la tabla, dividiéndola repetidamente por la mitad y decidiendo en qué mitad está el elemento requerido. Para que esto funcione, los valores de la tabla deben estar ordenados y en la definición de la tabla debe haberse especificado una cláusula ASCENDING o DESCENDING KEY.

Una de las principales características del COBOL que lo hacen tan adecuado para aplicaciones comerciales es la riqueza de facilidades para tratamiento de archivos. Todas las versiones de COBOL pueden manejar métodos de acceso secuencial, directo o indexado de una forma coherente y relativamente directa. Los archivos en dispositivos externos inicialmente se declaran en la sección de E/S (input-output section), el párrafo de control de ficheros (file-control) de la división de entorno (environment division). A cada archivo se le asigna un nombre de una sentencia SELECT independiente, que toma la forma:

Acceso indexado
La capacidad del COBOL de subdividir tipos de datos dentro de la división de datos (*data division*) permite al usuario construir complejos registros, que se pueden INDEXar y ser objeto de búsquedas (SEARCH). Además, el hecho de que la propia matriz y cada fila se declaren por separado significa que se pueden llevar a cabo MOVEs en bloque sobre los datos que encuadrarían a los diversos elementos constitutivos




```
SELECT nombre-archivo ASSIGN TO nombre-
dispositivo.
```

donde el nombre-archivo es un identificador del COBOL y el nombre-dispositivo es un identificador dependiente del sistema, que puede ser simplemente DSK o LPT, o un verdadero nombre de archivo del sistema. Si la versión de COBOL dada no requiere el nombre del archivo del sistema en este punto, en algún punto necesitará una cláusula adicional en la que poder especificar este nombre. Si en este punto no se da ninguna otra información, se asume que se trata de un archivo secuencial.

Existen numerosas cláusulas opcionales que permiten un control virtualmente completo (donde sea posible) de la forma en que realmente se almacena el archivo. Las dos más ampliamente utilizadas definen la organización (ORGANISATION) y el acceso (ACCESS). Hay tres opciones para ORGANISATION: SEQUENTIAL (secuencial, por defecto), RELATIVE (relativa, el nombre en COBOL para un archivo organizado por acceso directo mediante un número de registro) e INDEXED (indexada). Cuando la organización de un archivo es SEQUENTIAL, el único tipo de ACCESS permitido es SEQUENTIAL. Sin embargo, los otros dos tipos de organización también permiten el acceso RANDOM (aleatorio, directamente a un registro determinado) o DYNAMIC (dinámico), que es una combinación de RANDOM y SEQUENTIAL.

En el caso de archivos INDEXED, se debe declarar una RECORD KEY (clave de registro), que será uno de los campos del registro de datos utilizado como índice. En el caso de archivos RELATIVE, se debe declarar una RELATIVE KEY, que es un elemento de datos numérico empleado para almacenar el número de registro.

Cada nombre de archivo que se mencione en una sentencia SELECT se debe entonces definir en la sección de archivos (FILE SECTION) de la división de datos (DATA DIVISION), donde se da el nombre en

una declaración FD (*file definition*: definición de archivo) junto con otra información dependiente del sistema, como el tamaño del buffer, seguido por una definición de datos normal para la estructura del registro.

Aunque parezca un poco complicado declarar un archivo, las sentencias que se deben colocar en las divisiones de entorno (*environment*) y de datos (*data*) son estándares para la inmensa mayoría de las aplicaciones.

En la división de procedimientos (*procedure division*) hay varios verbos que se emplean específicamente para tratamiento de archivos. Se debe abrir cada archivo antes de utilizarlo:

```
OPEN nombre-archivo FOR modalidad-acceso.
```

donde la modalidad-acceso puede ser INPUT, OUTPUT o INPUT-OUTPUT; y, cuando se ha terminado con el archivo, debe ser cerrado:

```
CLOSE nombre-archivo.
```

La lectura y la escritura se realizan mediante los verbos READ (leer) y WRITE (escribir). Al igual que muchos verbos del COBOL, éstos tienen muchas opciones, incluyendo opciones para usar protección de registros y archivos en aplicaciones multiusuario. Sin embargo, en la mayoría de los casos sus usos son bastante directos; las formas básicas son:

```
READ nombre-archivo.
WRITE nombre-archivo.
```

La diferencia entre ambos reside en que READ requiere que el nombre del archivo y la ejecución de la sentencia llenen el registro de datos especificado para ese archivo en la sección de archivos (*file section*) de la división de datos (*data division*). La sentencia WRITE requiere el nombre de ese registro de datos y coloca su contenido en el dispositivo especificado en la sentencia SELECT.

Si el archivo se ha especificado con acceso secuencial, el READ leerá el siguiente registro del archivo y avanzará hacia el próximo. Se ha de leer la marca de fin del archivo para determinar si se ha llegado o no al final. La sentencia READ debe incluir la cláusula AT END, que especifica la acción a emprender al leer esta marca. La forma normal es utilizando un flag:

```
77 f-d-a PIC X VALUE 'N'.
88 fin-del-archivo VALUE 'S'.
............
PROCEDURE DIVISION.
PÁRRAFOS-PRINCIPALES-DE-CONTROL.
   OPEN INPUT in-archivo, OUTPUT out-archivo.
   READ in-archivo AT END MOVE 'S' TO f-d-a.
   PERFORM párrafo-procesar-registro
     UNTIL fin-del archivo.
PERFORM cerrar.
   STOP RUN.
PARRAFO-PROCESAR-REGISTRO.
............
   WRITE out-registro.
   READ in-archivo AT END MOVE 'S' TO f-d-a.
```

El WRITE escribirá el registro nuevo al final del archivo en cada ocasión.

Cuando el acceso al archivo es RANDOM o DYNAMIC, y la organización del archivo es INDEXED o RELATIVE, la lectura o escritura del archivo es un proceso con dos etapas. Primero, se debe colocar un valor adecuado en el campo de clave, y en el caso de un archivo RELATIVE, el número de registro requerido se coloca en la RELATIVE KEY antes de ejecutar la instrucción READ.

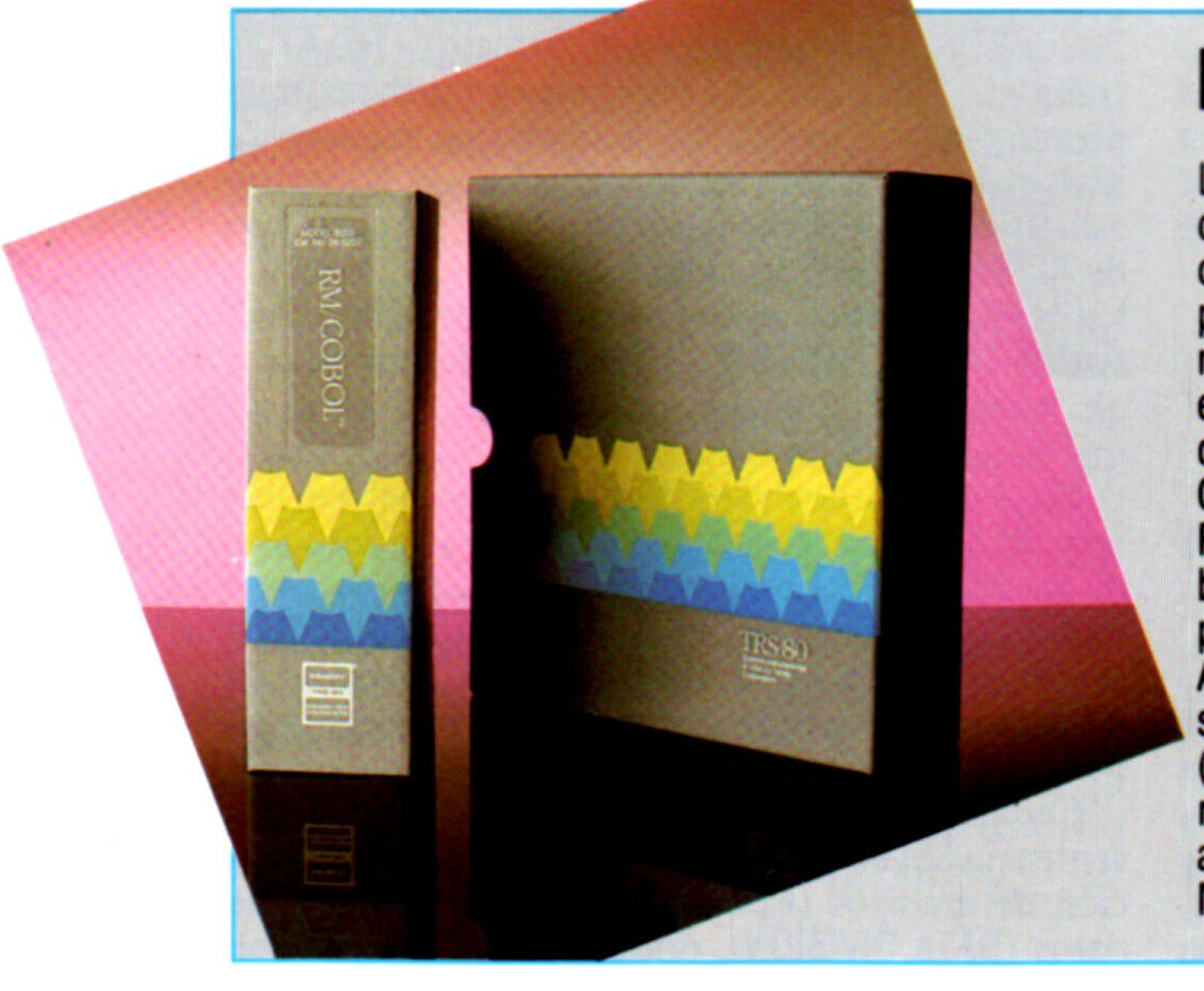
Los archivos INDEXED deben tener colocado un valor adecuado en el campo RECORD KEY. En lugar de la cláusula AT END, ha de haber una cláusula INVALID KEY que se ejecutará si no hay ningún registro que corresponda al valor clave dado.

En estos dos casos, el verbo WRITE sólo se utiliza para escribir registros nuevos. Un registro que se ha actualizado se escribirá utilizando el verbo REWRITE (reescribir), y se puede eliminar un registro utilizando DELETE (suprimir). Estos dos verbos requieren una cláusula INVALID KEY.

El acceso secuencial a un archivo siempre es posible utilizando:

```
READ nombre-archivo NEXT RECORD.
```

Ésta ha sido una introducción muy breve al tema del tratamiento de archivos en COBOL que, por ser una de sus facilidades más importantes, requeriría se le dedicara una serie completa. De hecho, aquí sólo hemos analizado brevemente la mayoría de las características del lenguaje.



La gama COBOL

La popularidad de que disfruta el COBOL en la comunidad empresarial trae aparejada la existencia de gran número de implementaciones del mismo para micros, aunque (debido a las limitaciones de memoria) muy pocas para micros personales. Es esencial, por supuesto, un sistema basado en disco. Algunas de las versiones más conocidas son CIS-COBOL de Microfocus, Microsoft COBOL y RM/COBOL (que vemos aquí). Todas ellas operan bajo CP/M y MS-DOS. Los usuarios de máquinas personales que ejecuten CP/M (Memotech, Amstrad y Commodore, p.ej.) podrían, por supuesto, utilizar cualquiera de estos paquetes (haciendo concesiones de memoria), pero podría resultarles caro. Al igual que con el FORTRAN, una alternativa más económica es Nevada COBOL, de NewStar Software

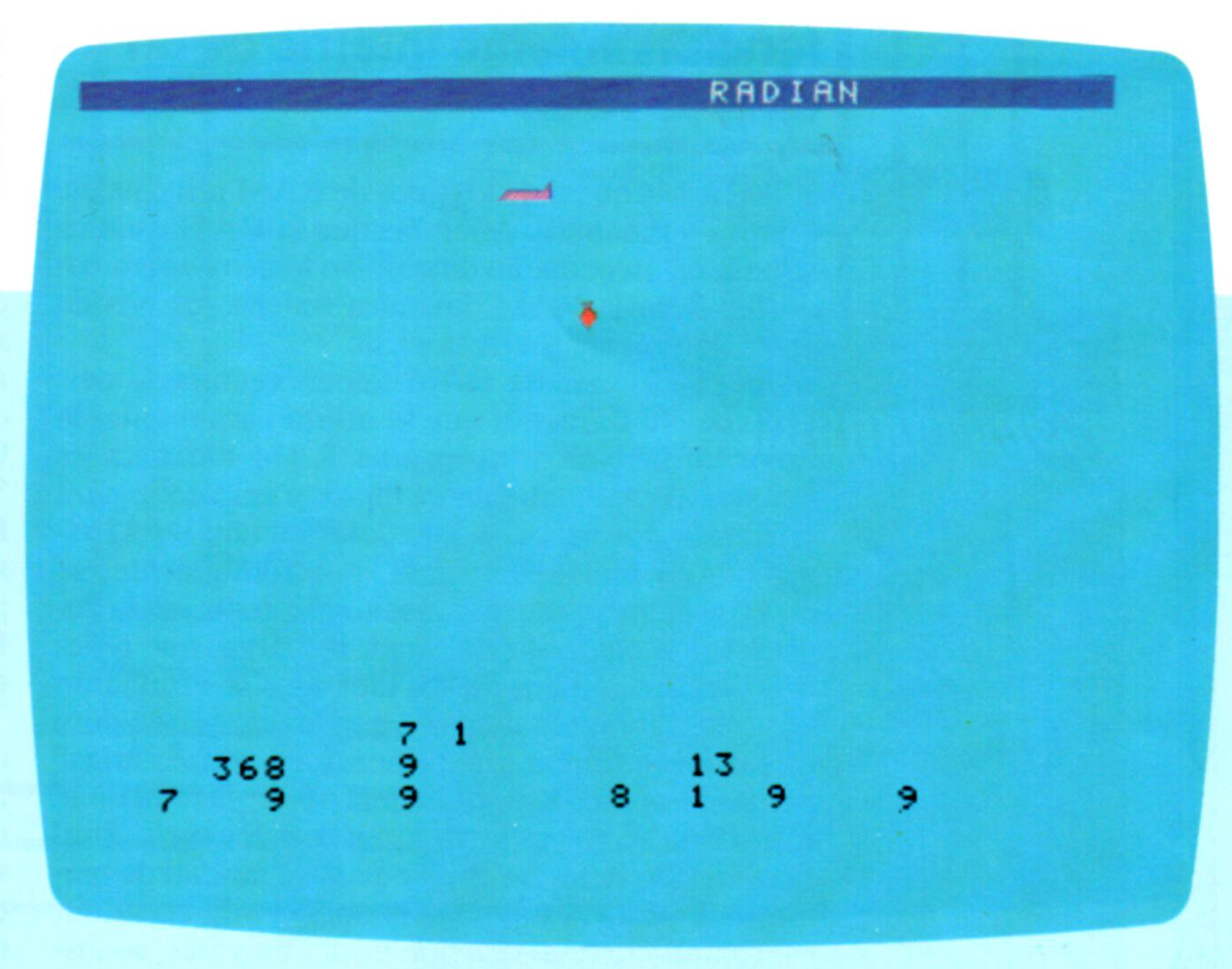
Liz Heaney



Numerix

Para los usuarios de un microordenador EXL 100, he aquí un atractivo programa de juegos escrito por Pierre Monsaut, en el estilo del conocido "Bombardeo aéreo"

En esta ocasión debe bombardear con ayuda de su avión las cifras que se encuentran en la parte inferior de la pantalla, a fin de añadir sus valores al total de puntos. Para soltar una bomba, pulse una tecla cualquiera. Cada cifra alcanzada aumenta el número de bombas disponibles.




```
100 REM ***********
110 REM * NUMERIX *
120 REM ***********
130 DIM TB(40,22)
140 R=0
150 GOSUB 850
160 GOSUB 560
170 LOCATE (AY,AX)
180 CALL COLOR("1MC")
190 PRINT A$;
200 CALL COLOR ("1RC")
210 CALL KEY1(D3,D4)
220 IF D3<>255 AND BY=0 THEN BX=AX:BY=AY
    +1:NM=NM-1
230 IF BY<>0 THEN BY=BY+1
240 IF BY>22 THEN LOCATE (BY-1,BX):PRINT N$;:BY=0:IF
    NM<1 THEN 410
250 IF BY<>0 AND TB(BX,BY)<>0 THEN GOSUB 310
260 IF BY<>0 THEN LOCATE (BY-1,BX):PRINT N$;:LOCATE
    (BY,BX):PRINT B$;
270 IF BY=0 THEN FOR I=1 TO 5:NEXT I
280 AX=AX-1
290 IF AX<1 THEN AX=38:LOCATE (AY,1):PRINT M$;
300 GOTO 170
310 LOCATE (BY-1,BX)
320 PRINT N$;
330 LOCATE (BY,BX)
340 PRINT N$;
350 S=S+TB(BX,BY)*10
360 TB(BX,BY)=0
370 BY=0
380 NM=NM+.5
390 GOSUB 720
400 RETURN
410 CALL COLOR("1BC")
420 LOCATE (10,15)
430 PRINT "PUNTOS:";S;
440 IF S>R THEN LET R=S
450 LOCATE (13,15)
460 PRINT "RECORD:";R;
470 LOCATE (16,15)
480 PRINT "OTRA?";
490 CALL KEY1(D3,D4)
500 IF D3<>255 THEN 490
510 CALL KEY1(D3,D4)
520 IF D3=255 THEN 510
530 IF D3<>78 THEN 150
540 CLS
550 END
560 CLS ("BCC")
570 N$=CHR$(32)
580 A$=CHR$(100)&CHR$(101)&N$
590 AX=38
600 AY=3
610 B$=CHR$(102)
620 M$=N$&N$&N$
630 BX=0
640 BY=0
650 GOSUB 810
660 FOR I=1 TO 15
670 GOSUB 720
680 NEXT I
690 NM=20
700 S=0
710 RETURN
720 J=INTRND(9)
730 X=INTRND(37)+1
740 Y=INTRND(3)+19
750 IF TB(X,Y)<>0 THEN 730
760 CALL COLOR("1BC")
770 LOCATE (Y,X)
780 PRINT CHR$(J+48);
790 TB(X,Y)=J
800 RETURN
810 CALL CHAR (100,"0000000000003F7FFF00")
820 CALL CHAR (101,"000000000103FFFFFF00")
830 CALL CHAR (102,"002810387C7C7C381000")
840 RETURN
850 FOR I=1 TO 40
860 FOR J=18 TO 22
870 TB(I,J)=0
880 NEXT J
890 NEXT I
900 RETURN
```

Rendimiento dinámico

Analizaremos la RAM dinámica e indagaremos en el funcionamiento interno de un chip de RAM

Ya sabemos que hay dos tipos de RAM principales: estática y dinámica. Mientras que la RAM estática se basa en un pequeño dispositivo lógico denominado flip-flop, la RAM dinámica retiene sus bits de datos como cargas electrónicas.

Ambas formas de RAM tienen ventajas y desventajas. El transistor que se utiliza para retener la carga de un único bit en una RAM dinámica es mucho más pequeño que el flip-flop empleado para retener la misma unidad de datos en una RAM estática. Las RAM dinámicas, por consiguiente, se pueden empaquetar de forma más densa en la superficie del chip. Sin embargo, las cargas que retienen los datos en una RAM dinámica se esfumarán cada pocos milisegundos y, por tanto, se requiere un sistema de circuitos adicional para "refrescarlas" periódicamente. Este problema no se plantea en el caso de la RAM estática y, en consecuencia, si el sistema sólo requiere una pequeña cantidad de memoria, la RAM estática es la opción más económica. Cuando se requieren memorias mayores, se justifica el gasto adicional que representa el sistema de circuitos de refresco y es más probable que se utilice la RAM dinámica.

Anteriormente hemos visto que la RAM estática normalmente se organiza en registros de ocho bits, teniendo cada chip ocho patillas de datos unidas a las ocho líneas del bus de datos. Las RAM dinámicas tienden a construirse en líneas diferentes; cada chip de RAM dinámica suele representar uno de los ocho bits de datos que componen una posición de la memoria, y ocho de tales chips, cableados en paralelo, constituyen los bytes de memoria.

En el primer diagrama vemos el 4116, con las conexiones de patillas típicas de un chip de 16 Kbits. En la actualidad muchos micros de ocho bits utilizan RAM dinámicas 4164 (64 Kbits) de este tipo para conseguir una memoria RAM de 64 Kbytes. Aunque podemos imaginarnos que el chip tiene un bit de "ancho" y 16 por 1 024 bits de "largo", en realidad está dispuesto en 128 filas por 128 columnas.

Si comparamos las patillas de salida de este chip con las de la ROM estática, podemos apreciar varias diferencias notables. En primer lugar, en lugar de ocho patillas de datos hay sólo dos, denominadas *data in* y *data out*. Cabría esperar que un chip de 16 Kbits requiriera 14 bits de direcciones para seleccionar cada bit de modo exclusivo, pero en realidad hay sólo siete. También hay presentes otras dos líneas hasta ahora desconocidas: $\overline{\text{RAS}}$ y $\overline{\text{CAS}}$, que son la *"strobe" de dirección de fila* y la *"strobe" de dirección de columna*, respectivamente. Estas dos señales de temporización permiten presentar la dirección en dos mitades (de allí las siete patillas de dirección en lugar de las 14 esperadas). La línea $\overline{\text{RAS}}$ también sirve para refrescar la RAM dinámica.

El proceso de refresco consiste en leer los datos de la RAM dinámica y escribirlos de nuevo para restablecer la carga. En nuestra RAM de ejemplo, esto se puede hacer de fila en fila. Por lo tanto, sólo se necesitan 128 operaciones para refrescar todo el chip. Si bien el refresco suele reducir la velocidad del procesador al demorar los accesos a la memoria durante los ciclos de refresco, es probable que la demora sea apenas del 5%.

Anteriormente hemos mencionado ya que los 14 bits de direcciones necesarios para una memoria de 16 Kbytes se pueden presentar en nuestro ejemplo de RAM dinámica en dos partes de siete bits. Esto se consigue a través de un chip lógico externo que junta las líneas de dirección de orden *low* y *high* con la señal de temporización $\overline{\text{CAS}}$. El segundo diagrama muestra una disposición simplificada en donde las líneas de direcciones de orden *low* (A0 a A6) están conectadas a las patillas de direcciones del chip cuando $\overline{\text{CAS}}$ está *high* y las líneas de direcciones de orden *high* están conectadas cuando $\overline{\text{CAS}}$ está *low*. De modo que en el chip de RAM propiamente dicho hay dos cerrojos (*latches*) que tienen la capacidad de "congelar" los datos que entran en ellos y, por lo tanto, la dirección de fila se toma cuando $\overline{\text{CAS}}$ está *high* y la dirección de columna se toma cuando $\overline{\text{CAS}}$ está *low*. La colocación de estos valores tratados con *latches* a través de decodificadores de 7 a 128 permite acceder al bit direccionado.

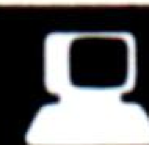
El último diagrama muestra cómo se conecta una memoria RAM dinámica de 16 Kbytes a las líneas de direcciones, de datos y de lectura/escritura del procesador. La línea $\overline{\text{CAS}}$ se conecta en paralelo a cada una de las ocho RAM y al chip lógico de direccionamiento. $\overline{\text{RAS}}$ y $\overline{\text{WE}}$ también están conectadas a las ocho RAM. Las patillas *data in* y *data out de* cada RAM están cableadas entre sí y conectan con una de las ocho líneas del bus de datos.

RAM dinámica

+12 V
DATA IN
$\overline{\text{WE}}$
$\overline{\text{RAS}}$
MA0
MA2
MA1
−12 V
RAM DINAMICA 4116
MASA
$\overline{\text{CAS}}$
DATA OUT
MA6
MA3
MA4
MA5
+5 V

Esta RAM de 16 Kbits posee conexiones de patillas típicas. A diferencia de una RAM estática, una RAM dinámica normalmente corresponde a un único bit de la palabra de datos, de ahí las líneas *data in* y *data out* individuales. Sólo está presente la mitad de las líneas de dirección que cabría esperar, simplificando considerablemente el cableado de tales RAM

Mike Clowes



Cambiando direcciones

A13, A6, A12, A5, A11, A4, A10, A3, A9, A2, A8, A1, A7, A0

B6, A5, B5, A4, B4, A3, B3, A2, B2, A1, B1, A0, B0

CHIP LÓGICO

MA6, MA5, MA4, MA3, MA2, MA1, MA0

$\overline{CAS}$

A7, A0, MA0, $\overline{SELA}$

CERROJO DE DIRECCIÓN DE COLUMNA

A7-A13

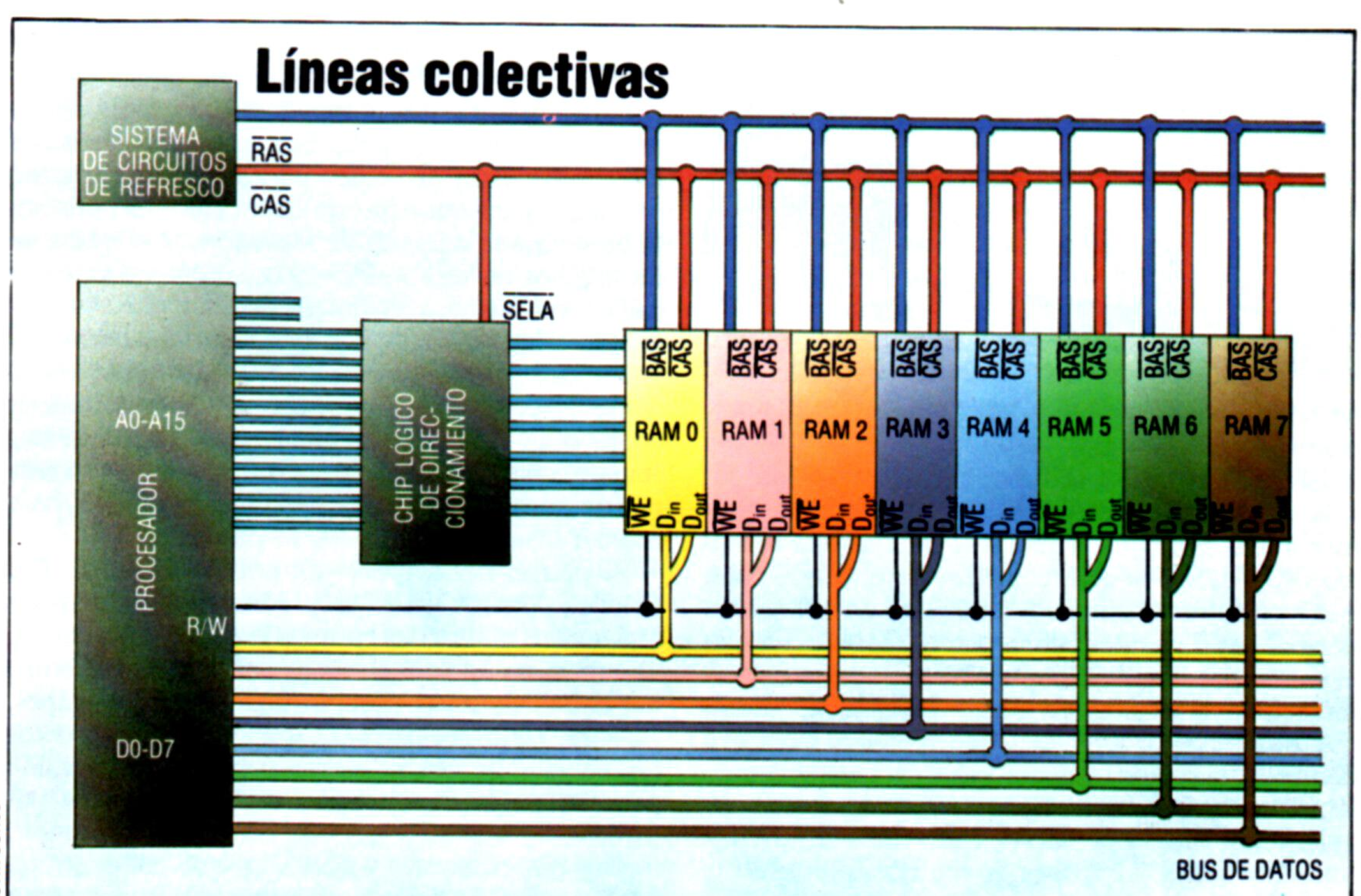
DECODIFICADOR DE 7 A 128

BIT SELECCIONADO

A0-A6

DECODIFICADOR DE 7 A 128

CERROJO DE DIRECCIÓN DE FILA



Líneas compartidas
Las 14 líneas de dirección necesarias para seleccionar cada bit en una RAM de 16 Kbits se utilizan para producir direcciones de columna y fila de siete bits. Para simplificar el sistema de circuitos, el chip propiamente dicho posee sólo siete patillas de dirección y la conmutación de las partes de orden *low* y *high* de la dirección de entrada normalmente las manipula un chip lógico externo (utilizando $\overline{CAS}$ como señal de sincronización). El diagrama muestra la sencilla disposición de lógica combinatoria necesaria para conmutar las dos mitades de la dirección

Figuras de ocho
Se puede construir una memoria de 16 Kbytes cableando ocho chips 4116 en paralelo de modo que compartan las líneas de dirección, R/W, $\overline{RAS}$ y $\overline{CAS}$. En dicha disposición, cada RAM dinámica estaría conectada a un único bit del bus de datos

Baraja informatizada

Iniciamos un nuevo proyecto en que crearemos un programa para jugar a un conocido juego de cartas: el veintiuno

Los juegos de naipes son ideales para implementarlos en un ordenador. Sus reglas de juego y estrategias, rígidamente definidas, con frecuencia se basan en gran medida en el análisis matemático. El veintiuno, por ejemplo, es especialmente fácil de programar y las reglas son muy directas, lo que hace que codificarlo para el ordenador sea una tarea sencilla. Sin embargo, los principios de la creación de una baraja de naipes, la visualización de éstos y la evaluación de las manos, ilustran ampliamente los tipos de problemas que suelen plantearse.

En primer lugar, encarguémonos de preparar la baraja y visualizar los naipes individuales, incorporando listados separados para el Commodore 64, el Spectrum, el BBC Micro y la gama Amstrad. Sin embargo, por el momento sólo ofreceremos la versión Commodore.

Una baraja

DP SEÑALA LA PARTE SUPERIOR DE LA BARAJA

NÚMERO DE NAIPE

13	10	5	7	1	2	7	9	11	13	...	5	6	5
2	1	3	2	4	2	3	1	3	1	...	1	2	4

DK(,)

NÚMERO DE PALO

La matriz bidimensional DK(,) se utiliza para retener la baraja de naipes utilizada en el juego. Muchos juegos de naipes sencillos reparten las cartas al azar, pero un aspecto importante del arte de jugar al veintiuno es recordar en qué lugar del mazo se encuentran determinados naipes. Utilizando una matriz podemos simular la acción del reparto con una baraja verdadera y la devolución de los naipes usados a la parte de abajo con mayor precisión; se utiliza un puntero, DP, para indicar dónde está el "tope" actual de la baraja

Kevin Jones

El método más sencillo de representar una baraja de naipes es retenerla como una matriz bidimensional. En nuestro juego se DIMensiona la matriz DK(,) de modo que tenga 52 elementos de largo y dos elementos de ancho. Dado que en una baraja normal hay 52 naipes (sin contar los comodines), es evidente por qué necesitamos 52 elementos en una dirección. Tomando un naipe individual, vemos que posee dos propiedades que lo hacen exclusivo dentro de la baraja: su valor (as, 2, 3, etc.) y su palo (corazones, diamantes, tréboles y picas). Por consiguiente, el valor y el palo de cada carta se retienen separadamente dentro de la matriz, utilizando la segunda dimensión.

Para que la matriz pueda ser numérica, hemos empleado los números del 1 al 13 para retener el valor del naipe, de modo que 1 es el as y 13 es el rey, y un número del 1 al 4 para representar el palo. La matriz de la baraja y otras matrices utilizadas para representar los naipes y sus posiciones se deben inicializar al comienzo del programa, lo que se realiza mediante la subrutina de la línea 500.

El método que hemos empleado para simular la barajada de naipes utiliza un par de bucles anidados. El bucle exterior va contando a través de los cuatro palos y el bucle interior va contando a través de los 13 naipes de cada palo. Por tanto, las dos variables contadoras de los bucles representan el palo del naipe y el valor del naipe a entrar en la matriz de la baraja dentro del bucle interno. Todo cuanto necesitamos hacer es seleccionar el punto en el cual deseamos colocar el naipe actual dentro de la baraja.

Puesto que se supone que el mazo está barajado, nuestra rutina selecciona el punto de entrada a éste de forma aleatoria; pero aquí hay un pequeño obstáculo: es posible que la posición seleccionada en la matriz de la baraja ya esté ocupada. Para sortear este problema, una pequeña rutina dentro del bucle interno comprueba el punto de entrada seleccionado al azar: si no está vacío, incrementa el punto de entrada (dando toda la vuelta hasta la parte de arriba de la baraja si fuera necesario) hasta hallar un espacio vacío. Al salir de nuestra estructura de bucles anidados, nuestra baraja contendrá 52 naipes distintos dispuestos por un orden aleatorio.

Para visualizar un naipe, todo cuanto necesitamos es su valor y su palo. Sin embargo, hemos de interpretar estos dos datos para producir el patrón del naipe y las etiquetas de los ángulos del mismo. Si bien para la mayoría de los naipes la etiqueta de los ángulos corresponderá directamente al valor de cada carta, el As, el Valet, la Dama y el Rey necesitarán etiquetas A, J, Q y K. Asimismo, hemos de hacer una pequeña trampa representando el 10 como T, para que la etiqueta se pueda visualizar como una única columna en el naipe. La forma más sencilla de manejar todo esto es preparando una matriz en serie de números de tarjeta, CN$(), para retener las etiquetas de los 13 naipes.

El patrón del naipe es un poco más difícil. Por razones de simplicidad, representaremos todas las imágenes de las cartas como si fueran ases, es decir, visualizando un solo símbolo del palo en el centro del naipe. Si usted observa una baraja de naipes, verá que todos los patrones son formas geométricas regulares y, de hecho, es bastante fácil diseñar una plantilla en la cual tengan cabida todos los patrones.

Hay tres columnas y siete filas que pueden retener símbolos de palos, lo que nos da 21 posiciones

de símbolo en total, y hay varios métodos de retener patrones. Aquí hemos optado por retener cada patrón de naipe como una serie binaria de 21 elementos, usando 1 para representar un símbolo visualizado y 0 cuando no haya presente ningún símbolo. Las definiciones para los 13 naipes están retenidas en sentencias de datos en la línea 2100 y se leen en la matriz CD$(). Ahora podemos analizar la rutina de visualización de naipes en sí misma.

En la versión para el Commodore 64 que ofrecemos aquí no hay ningún método de situar directamente el cursor en un punto dado de la pantalla. Por lo tanto, el programa utiliza una subrutina en la línea 900 para posicionar el cursor en un punto definido por TX y TY.

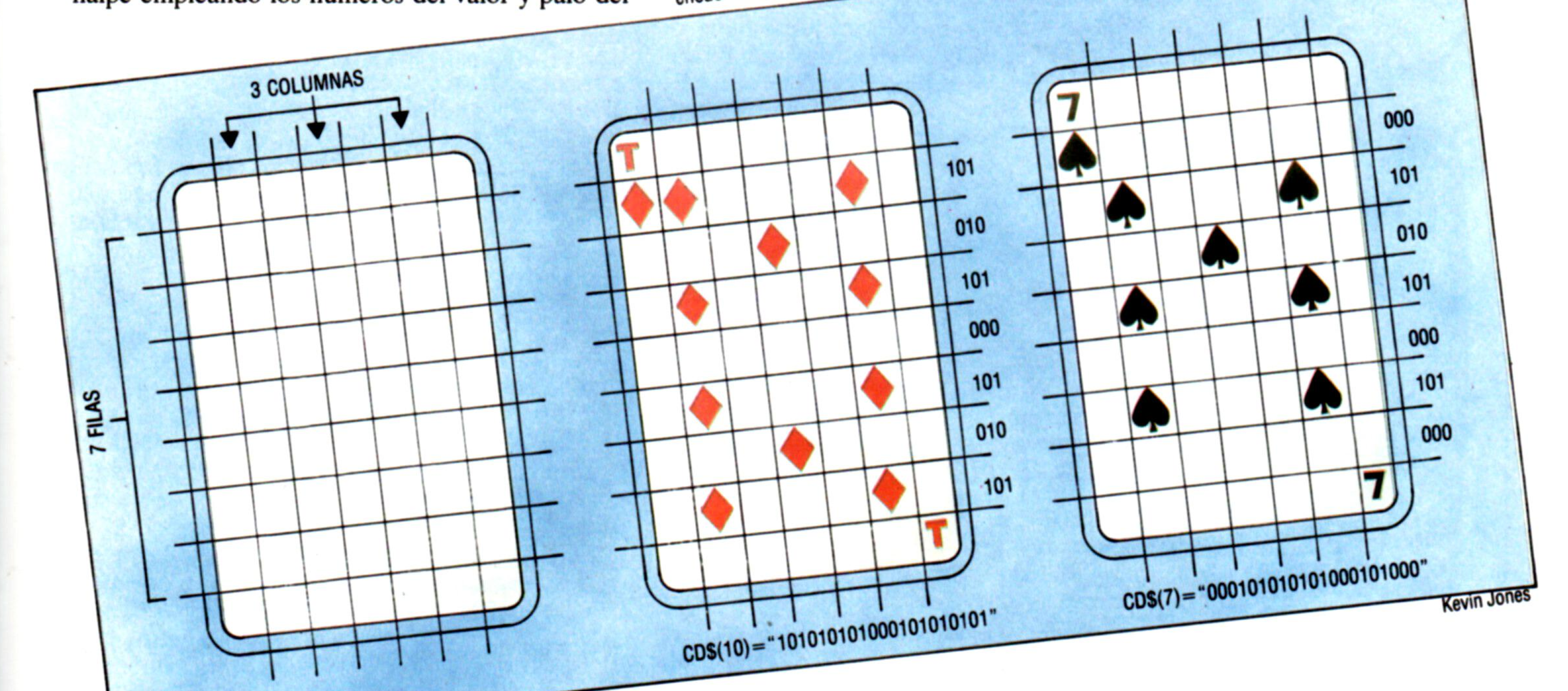
La subrutina de la línea 1050 imprime la silueta del naipe usando caracteres predefinidos. Luego la subrutina de la línea 1100 añade los detalles del naipe empleando los números del valor y palo del mismo, que se han pasado como CN y SU. La primera tarea es seleccionar el símbolo del palo de una serie de los cuatro símbolos definida en la rutina de inicialización. El color del naipe también se puede hallar fácilmente comprobando el número del palo. Puesto que los palos están dispuestos por orden de corazones, diamantes, tréboles y picas, simplemente necesitamos establecer el color del naipe en negro si el número de palo es mayor que dos y, de lo contrario, establecerlo en rojo.

Ahora se puede imprimir el patrón a partir de la descripción binaria retenida en CD(CN). Se utilizan un par de bucles anidados para recorrer cada una de las siete filas, tomando la serie binaria en grupos de tres y ensamblando una serie de espacios y símbolos de palo que constituyen la fila actual con la que se está trabajando. Tras imprimir el patrón del naipe, podemos añadir las etiquetas de los ángulos posicionando el cursor en el ángulo e imprimiendo CN$(CN).

En esta etapa veremos cómo se posicionan los naipes. Durante el juego, se les repartirán cartas al jugador o al ordenador y se las colocará de forma tal que queden superpuestas pero desplazadas en dos unidades horizontalmente y una unidad verticalmente. Para llevar el registro de la posición en la cual se ha de imprimir el naipe siguiente, se utilizan las matrices X() e Y(). El primer elemento retiene la posición del siguiente naipe para la mano del jugador y el segundo la posición del siguiente naipe para el ordenador. De este modo, al comenzar la rutina de impresión de naipes el cursor se desplaza hasta X(PL), Y(PL), donde PL es el número de jugador (1 o 2). Al final de la rutina, X(PL) e Y(PL) se incrementan en 2 y 1, respectivamente, listas para repartir el siguiente naipe a ese jugador.

Creación de una carta

Los patrones para los 13 tipos de naipe se almacenan como series binarias. Utilizando una plantilla de tres columnas por siete filas, nuestro diagrama ilustra cómo se han creado las series binarias para un patrón de "diez" y "siete". La rutina de visualización de naipes trabaja a través de la serie binaria adecuada en grupos de tres, imprimiendo el símbolo del palo determinado cuando se encuentra un uno y un espacio cuando se encuentra un cero



En la versión del juego que hemos programado, el ordenador juega como si fuera la banca, de modo que el primer naipe se le reparte boca abajo. Por consiguiente, necesitamos una pequeña rutina para visualizar el reverso de esta carta. La subrutina de la línea 1200 maneja esto llamando primero a la rutina de silueta de naipes y rellenando luego el interior utilizando un carácter de tablero.

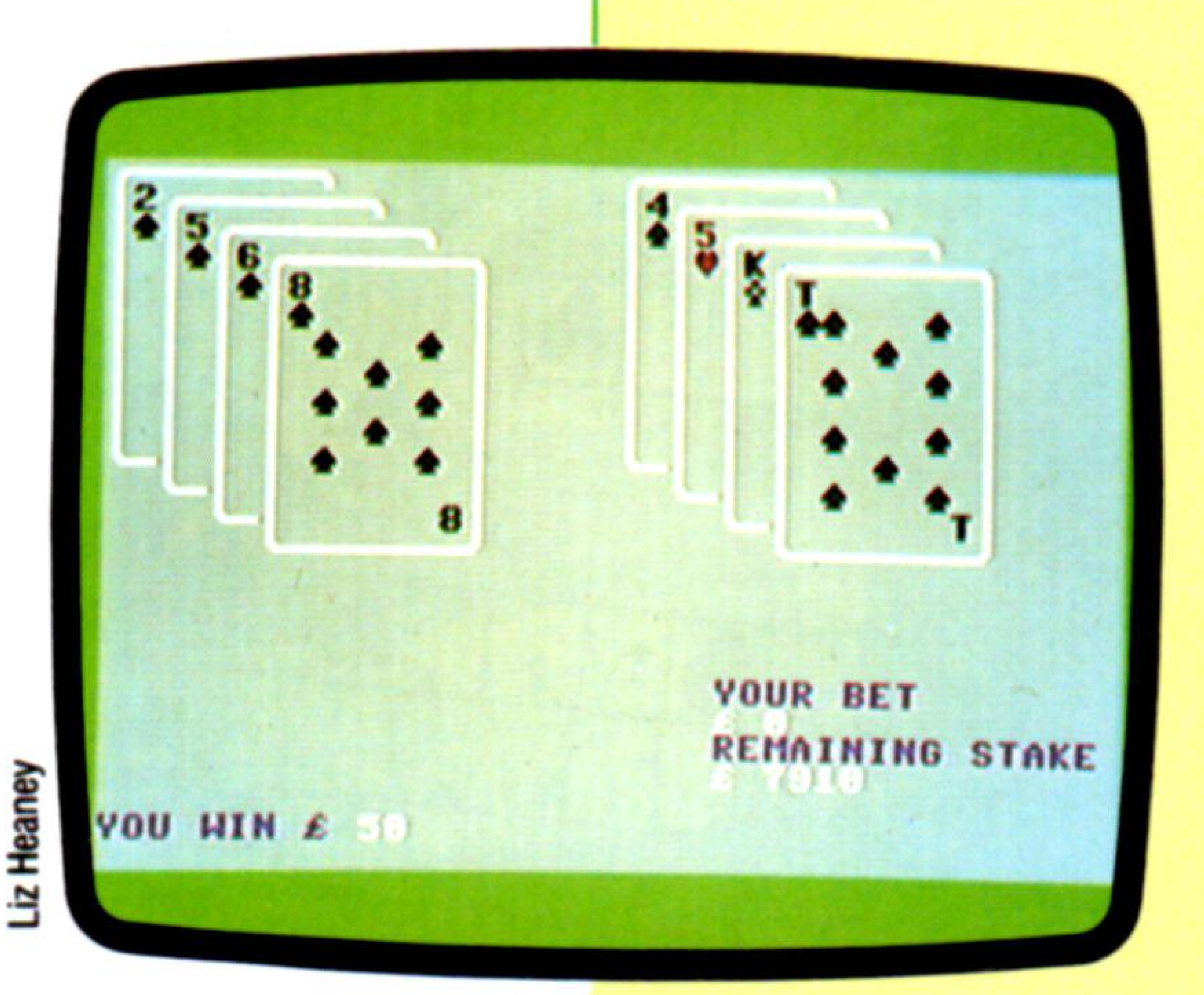
La última rutina de esta sección nos permite repartir naipes de la parte de arriba de la baraja y visualizarlos. La breve subrutina de la línea 1300 es la encargada de esto. Toma los elementos de la matriz de la baraja que corresponden a la parte de arriba del mazo y los coloca en CN y SU, lista para llamar a la rutina de visualización de naipes.

La forma más obvia de manipular el reparto sería tomar los primeros elementos de la matriz de la baraja, desplazar todos los otros elementos un lugar hacia adelante y colocar los elementos que se acaban de quitar en DK(52,1) y DK(52,2). En realidad,

todo este movimiento de elementos de la matriz lleva mucho tiempo y, en última instancia, es innecesario. En cambio, podemos utilizar una variable, DP, que señale hacia el "tope" del mazo y se incremente cada vez que se reparte un naipe, pasando de la posición 52 a la 1 si fuera necesario. Para repartir un naipe, por lo tanto, tomamos el elemento DK(DP,1) como el número del naipe y DK(DP,2) como el número de palo.

Al objeto de ver el efecto de nuestro trabajo hasta este momento, se pueden añadir las siguientes líneas para llamar a las diversas rutinas de modo que tanto al jugador como al ordenador se les repartan cinco cartas:

```
60 PL=1:FOR C=1 TO 10
70 PL=3−PL:REM NUMERO DEL JUGADOR
80 FL=0:GOSUB 1300:REM REPARTIR NAIPE
90 INPUT'PULSAR RETURN PARA
   SIGUIENTE NAIPE';RESP$
100 NEXT C
```



Liz Heaney

Necesitas manos...
Nuestro programa de veintiuno visualiza los naipes de las manos del jugador y de la banca en las mitades izquierda y derecha de la pantalla. El desplazamiento hacia un lado y hacia abajo de cada naipe recién repartido permite ver todos los valores de los naipes de cada mano. En esta etapa del proyecto tan sólo se podrá visualizar los naipes. Los avisos, mensajes y visualizaciones de apuestas serán el tema de futuros capítulos

Visualización de naipes y barajada

Programa principal:

```
10 REM **** VEINTIUNO COMMODORE 64 ****
20 GOSUB 500:REM INIC MATRICES ETC
50 REM **** BUCLE DEL JUEGO ****
55 GOSUB 600:REM INIC JUEGO
```

Inicialización de matrices:

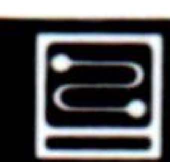
```
500 REM ***** INIC MATRICES ETC ****
510 SP$=" ":DW$=" ":FOR I= 1 TO 25:DW$=DW$+
    CHR$(17):SP$=SP$+ " ":NEXT I
511 SP$=SP$+LEFT$(SP$,14)
512 BK$=" ":LI$=" ":FOR I=1 TO 7:LI$=LI$+
    CHR$(195):BK$=BK$+ CHR$(166):NEXT I
513 BR$=CHR$(194)
515 DIM X(2),Y(2):REM POSICIONES SIGUIENTE NAIPE
520 SU$=CHR$(211)+ CHR$(218)+ CHR$(216)+
    CHR$(193):REM TIPOS DE PALO
530 DIM CN$(13):FOR I=1 TO 13:READ CN$(I):NEXT:REM
    LEER DATOS NUMEROS
540 DIM CD$(13):FOR I=1 TO 13:READ CD$(I):NEXT:REM
    LEER DATOS PATRONES
560 DIM DK(52,2):REM PREPARAR MATRIZ BARAJA NAIPES
570 GOSUB 3000:REM BARAJAR MAZO
580 POKE 53280,5:POKE 53281,15:REM COLORES PANTALLA
590 RETURN
600 REM **** INIC JUEGO ****
605 PRINT CHR$(147):REM BORRAR PANTALLA
620 X(1)=0:Y(1)=0:X(2)=20:Y(2)=20
630 RETURN
```

Rutina de visualización de naipes:

```
900  REM **** PRINT AT ****
910  PRINT CHR$(19);:PRINT
     LEFT$(DW$,TY);TAB(TX);:RETURN
1000 REM **** VISUALIZAR NAIPE ****
1010 GOSUB 1050:GOSUB 1100:RETURN
1050 REM **** NAIPE EN BLANCO ****
1055 TX=X(PL):TY=Y(PL):GOSUB 900:REM
     POSICION
1060 PRINT TAB(TX);CHR$(5);CHR$(213);LI$:CHR$
     (201)
1070 FOR I=1 TO 9:PRINT TAB(TX);BR$;" ";BR$:
     NEXT I
1080 PRINT TAB(TX);CHR$(202);LI$;CHR$(203)
1090 RETURN
1100 REM **** DETALLES NAIPE ****
1120 TX=X(PL)+ 2:TY=Y(PL)+ 2:GOSUB 900:REM
     POSICION
1125 CT$=MID$(SU$,SU,1):REM SELECCIONAR TIPO
     PALO
1127 CO$=CHR$(28):IF SU>2 THEN CO$=CHR$(144):REM
     SELECCIONAR COLOR
1128 PRINT CO$;
1130 FOR I=1 TO 19 STEP 3
1140 CC$=MID$(CD$(CN),I,3):CL$=" "
1142 FOR J=1 TO 3:C$=CHR$(29)+ CHR$(29)
1144 IF MID$(CC$,J,1)="1" THEN C$=CT$+
     CHR$(29)
1146 CL$=CL$+C$:NEXT J
1150 PRINT TAB(X(PL)+ 2);CL$:NEXT I
1160 REM **** AÑADIR ETIQUETAS ANGULOS ****
1170 TX=X(PL)+1:TY=Y(PL)+ 1:GOSUB 900:PRINT
     CN$(CN):REM NUMERO
1180 TY=TY+ 1:GOSUB 900:PRINT CT$:REM PALO
1190 TX=X(PL)+ 7:TY=Y(PL)+9:GOSUB 900:PRINT
     CN$(CN):REM NUMERO DE ABAJO
1192 X(PL)=X(PL)+ 2:Y(PL)=Y(PL)+ 1
1195 RETURN
1200 REM **** VISUALIZAR REVERSO NAIPE ****
1210 GOSUB 1050:GOSUB 1250:RETURN
1250 REM **** REVERSO NAIPE ****
1255 TX=X(PL):TY=Y(PL)+ 1:GOSUB 900:REM
     POSICION
1260 FOR I=1 TO 9:PRINT TAB(TX);BR$;CHR$(156);BK$ ;
     CHR$(5);BR$:NEXT I
1270 X(PL)=X(PL)+2:Y(PL)=Y(PL)+ 1
1280 RETURN
1300 REM **** REPARTIR UN NAIPE ****
1310 CN=DK(DP,1):SU=(DP,2)
1320 DP=D+ DP+ 1:IF DP> 52 THEN DP=1:REM BARAJA
     CIRCULAR
1330 IF FL=1 THEN GOSUB 1200:RETURN:REM VISUALIZAR
     REVERSO NAIPE
1335 GOSUB 1000:RETURN:REM VISUALIZAR NAIPE
2000 :REM **** DATOS NUMEROS NAIPES ****
2010 DATA A,2,3,4,5,6,7,8,9,T,J,Q,K
2100 REM **** DATOS VISUALIZACION NAIPES ****
2110 DATA"000000000010000000000":REM A
2120 DATA"000010000000000010000":REM 2
2130 DATA"000010000010000010000":REM 3
2140 DATA"000101000000000101000":REM 4
2150 DATA"000101000010000101000":REM 5
2160 DATA"000101000101000101000":REM 6
2170 DATA"000101010101000101000":REM 7
2180 DATA"000101010101010101000":REM 8
2190 DATA"101000101010101000101":REM 9
2200 DATA"101010101000101010101":REM 10
2210 DATA"000000000010000000000":REM J
2220 DATA"000000000010000000000":REM Q
2230 DATA"000000000010000000000":REM K
```

Mezclando la baraja:

```
3000 REM **** BARAJAR EL MAZO ****
3005 R=RND(−TI):DP=1
3007 FOR I=1 TO 52:DK(I,1)=0:NEXT I
3010 FOR I=1 TO4:FOR J=1 TO 13
3020 EP=INT(RND(1)* 52)+ 1:REM SELECCIONAR PUNTO
     ENTRADA
3030 IF DK(EP,1)=0 THEN 3050
3040 EP=EP+ 1:IF EP> 52 THEN EP=1
3045 GOTO 3030
3050 DK(EP,1)=J:DK(EP,2)=I:NEXT J,I
3060 RETURN
```



La dirección exacta, por favor

Investigaremos los modos de direccionamiento del 68000. Esto nos ayudará a la hora de programar el chip con registros y tablas

En el capítulo de introducción de esta serie hablamos de la capacidad de direccionamiento del 68000 a propósito del juego de instrucciones. En particular advertimos que a pesar de tener un amplio abanico de modos de direccionamiento con los que es fácil referenciar bytes, palabras y palabras largas, debemos ser muy cautos en el empleo de tales modos con determinadas instrucciones.

Sea la siguiente *instrucción generalizada*:

OPCODE fuente, destino

Ésta es una manera semiformal de describir lo que realiza una clase de instrucciones con los operandos fuente y destino. La secuencia puede que exija tomar el operando fuente (dondequiera que se encuentre, y a través del cálculo que sea preciso para llegar hasta él), realizar la operación definida por el opcode sobre el operando y depositar el resultado en el operando destino (aquí también puede que sea preciso realizar algún cálculo para obtener la dirección de este operando). Por ejemplo, la instrucción MOVEA D3,A6 hace que el contenido de D3 (fuente) se lleve a A6 (destino). El opcode, MOVEA, indica que ha de moverse una dirección.

Se trata de un ejemplo muy sencillo de direccionamiento, donde no es preciso cálculo alguno para obtener la dirección de los operandos. En el otro extremo podríamos encontrarnos con que uno de los operandos esté direccionado mediante la suma del contenido de un registro de direcciones y un desplazamiento entero más el contenido de un registro índice (similar a la instrucción LD r, (IX+d) del Z80). Ya tendremos tiempo de comentar esto más adelante; de momento, baste con advertir que puede haber un buen puñado de operaciones aritméticas por realizar hasta dar con la dirección de los operandos.

Volviendo a nuestro modelo generalizado de instrucciones, también es posible que el opcode precise tan sólo un operando, como en este caso:

OPCODE fuente

Por ejemplo, la instrucción de bifurcación (como BRA BACKHERE) sólo necesita un operando (es decir, la dirección para bifurcar hacia BACKHERE). Finalmente, podemos también encontrarnos sólo con:

OPCODE

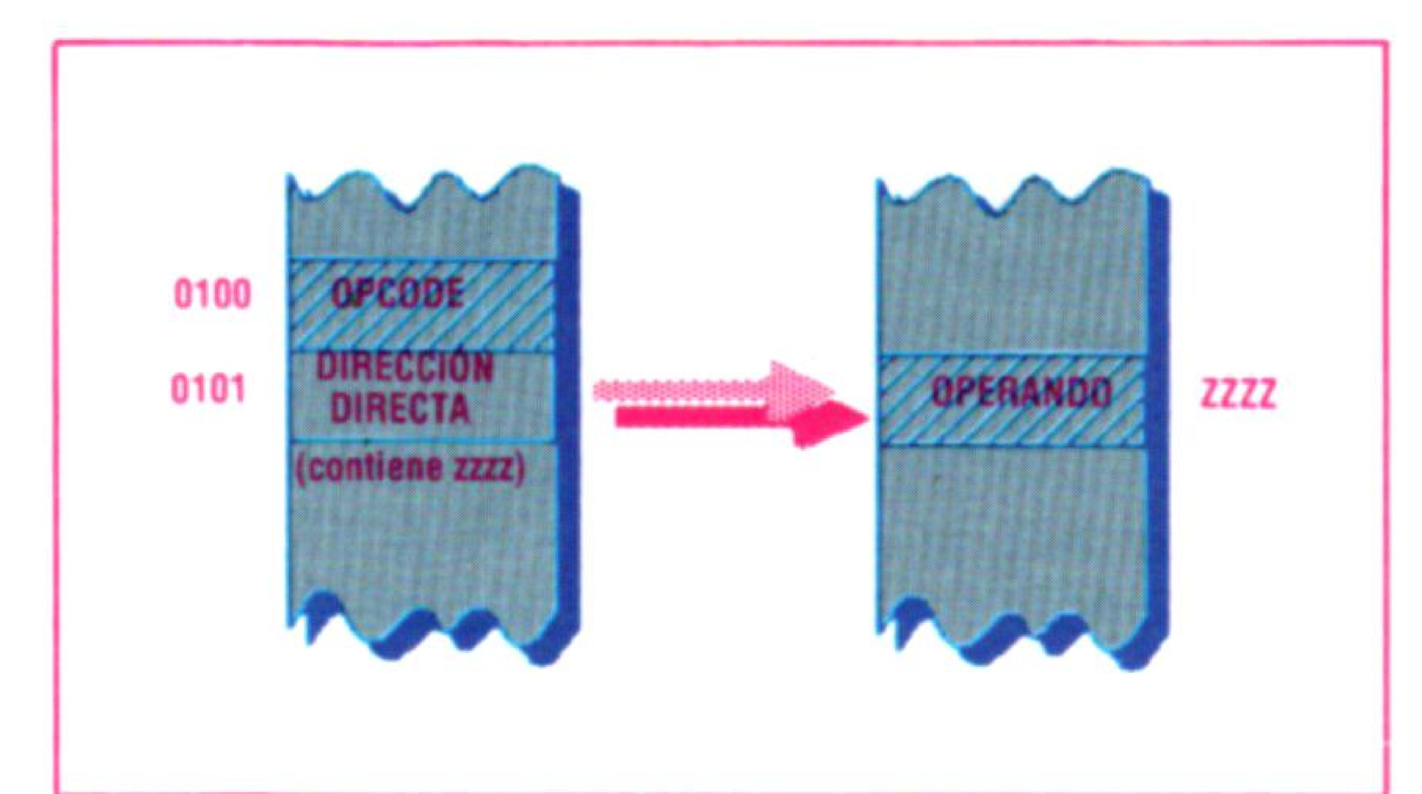
sin operando alguno. Un ejemplo típico de esta forma es la instrucción NOP, que sirve para indicar que no se realice ninguna operación. En efecto, se trata de una instrucción ficticia, y su utilidad es evidente en los parcheos manuales o en el cambio de programa en la memoria también manual.

Al emplear este modelo generalizado de la gama de instrucciones lo importante es tener en cuenta que siempre que haya operandos habrá algún tipo de cálculo de direcciones. Este cálculo es el que ha de ser especificado por el programador entre el conjunto de los cálculos o los modos de direccionamiento disponibles en el 68000.

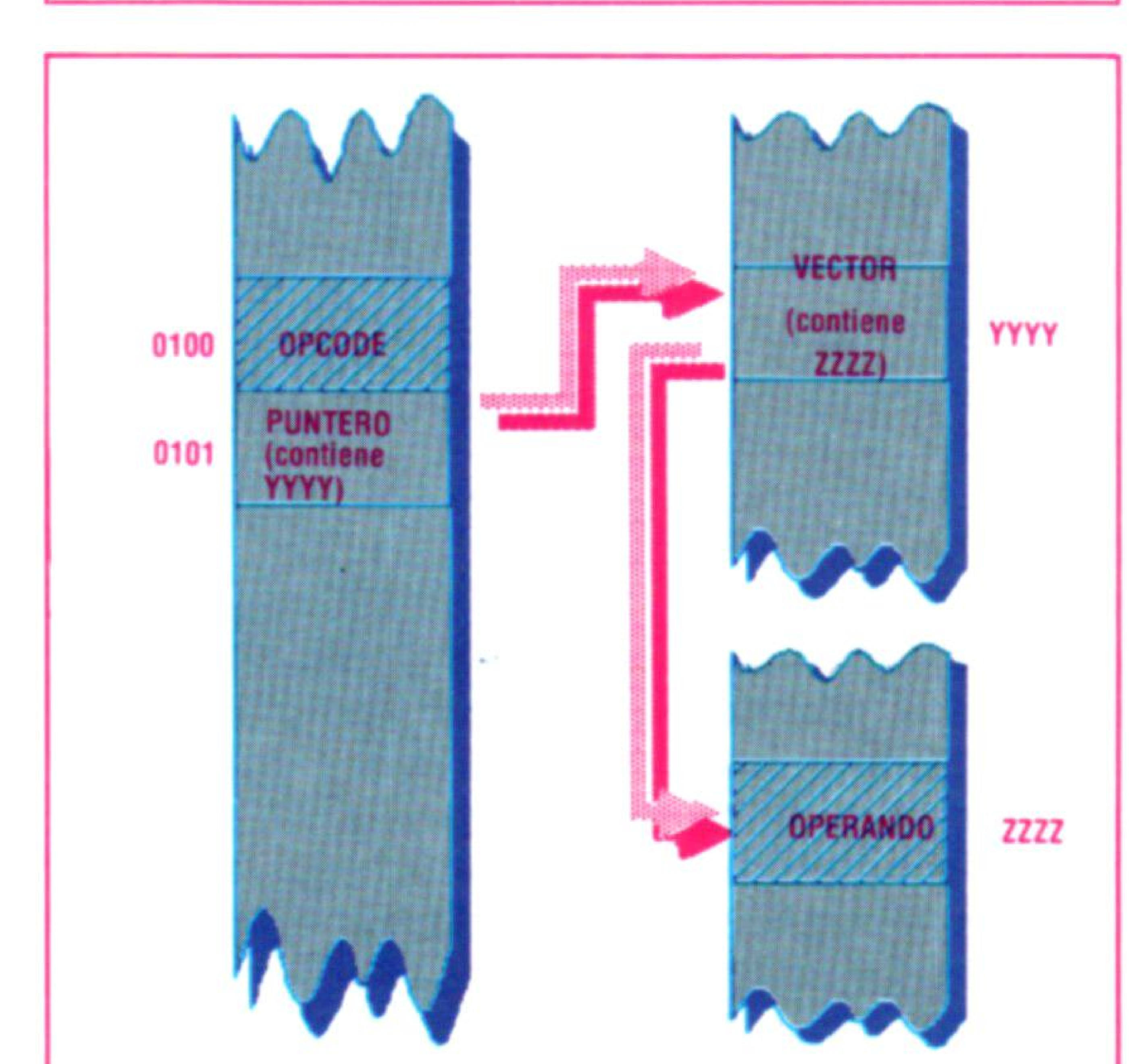
Muchos ordenadores disponen de al menos cinco *modos de direccionamiento* o maneras diferentes de direccionar los operandos. Nuestros diagramas ilustran la diferencia operativa de dos de estos modos:

- *Direccionamiento directo (o absoluto)*: En este modo la dirección de memoria del operando se almacena en la propia instrucción.
- *Direccionamiento indirecto (o puntero)*: En la instrucción se da una dirección de memoria que contiene a su vez la dirección del operando.

Como muestra el dibujo, en el direccionamiento directo el opcode opera sobre el operando que está en la dirección XXXX, mientras que en el direccionamiento indirecto, el operando será hallado en la po-



Direccionamiento directo
En este modo de direccionamiento el opcode (código de operación) de la instrucción es seguido inmediatamente por la dirección de una posición de memoria que contiene los datos del operando



Direccionamiento indirecto
El direccionamiento indirecto requiere que la operación especificada por el opcode acceda a su operando mediante una dirección "vector". La dirección del vector sigue inmediatamente al opcode, y el vector contiene la dirección del operando requerido. Este modo es muy útil cuando se calcula la dirección de un operando en tiempo de ejecución, dado que sólo se necesita calcular de nuevo el contenido del vector

sición ZZZZ, a la cual apunta un *puntero* que en este caso se encuentra en la posición YYYY de la memoria. Los otros tres modos principales de direccionamiento son:

- *Modo inmediato*: Cuando uno de los operandos de la instrucción es una constante. Por ejemplo, en la instrucción MOVEQ #25,D3 la constante 25 está direccionada en modo inmediato.
- *Modo de registros*: En este modo, el operando es uno de los registros disponibles y es especificado en el mismo código de la instrucción. Los operandos en la siguiente instrucción MOVE D2,D4 son los registros de datos D2 y D4.
- *Modo implícito*: Aquí los operandos están implícitos en la misma instrucción. Por ejemplo, en el caso de RTS (R*e*T*urn from* S*ubroutine*: retorno de subrutina) son operandos implícitos el puntero de la pila y el contador del programa.

Observemos con mayor atención la manera en que el 68000 direcciona sus operandos. Además de los modos generales de direccionamiento que acabamos de ver, el 68000 tiene un *modo relativo de contador de programa* (también llamado *PC relativo*). Examinemos estos modos uno por uno:

- *Direccionamiento absoluto*: Empleando este modo podemos acceder a cualquier posición de la memoria. La dirección del operando aparece después de la instrucción. Por ejemplo:



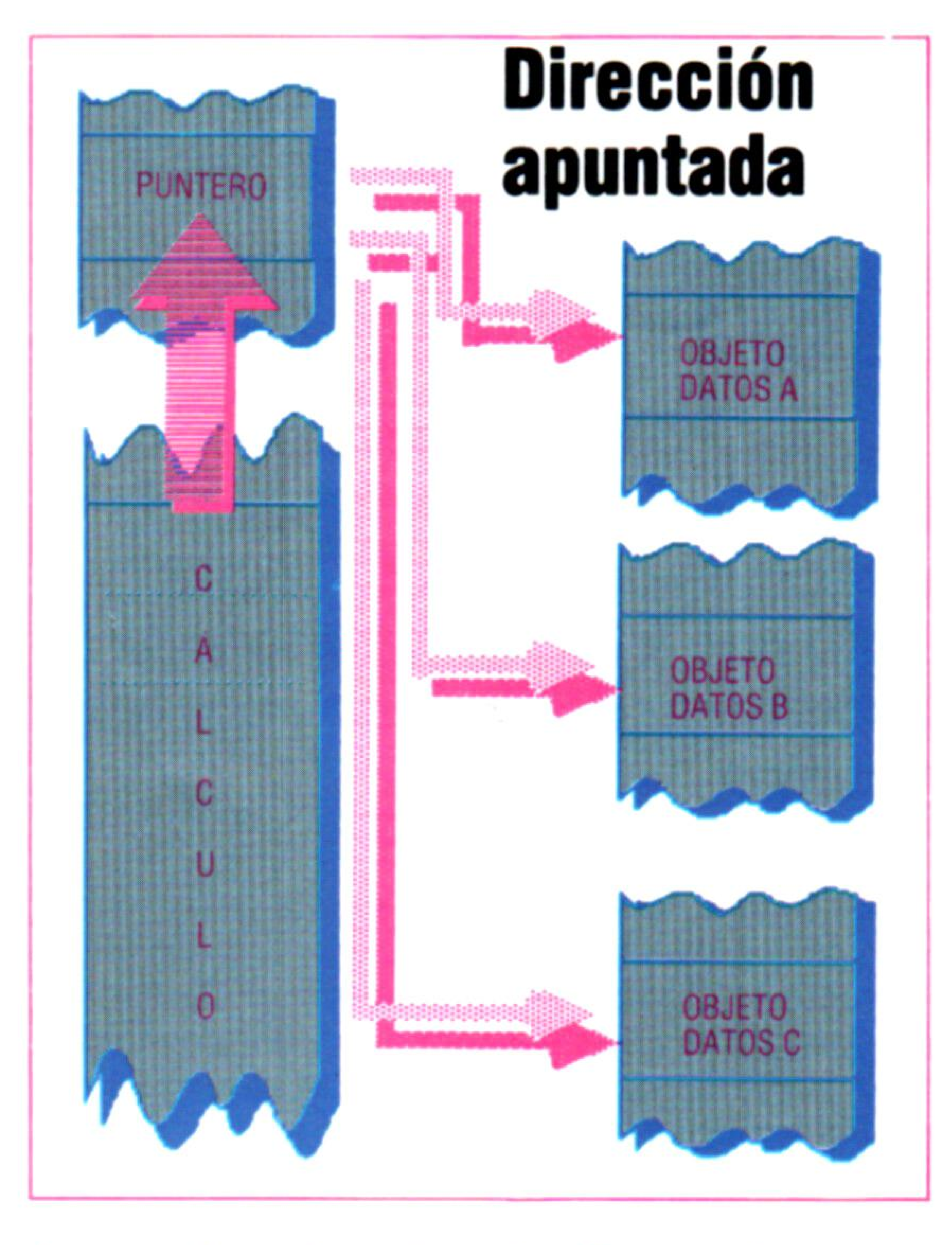
Direc.:	Código:	Etiq.:	Instrucción:	
1000	D678		ADD	DATA,D3
1002	2000			
.				
.				
.				
2000	0001	DATA	DC.W	1

En este ejemplo podemos ver que el nombre simbólico DATA ha recibido la dirección $2000, la instrucción "suma (ADD) la fuente absoluta (DATA) al destino D3" ha sido codificada como D678, y que DATA de dirección absoluta se alberga en la posición $1002 (llamada *extensión de la palabra*). Otro ejemplo donde sólo encontramos un operando es:

Dir.:	Código:	Etiq.:	Instrucción:	
1000	4278		CLR	COUNT
1002	3000			
.				
.				
.				
3000	0	COUNT	DS	1

Aquí el contenido de la posición $3000 (COUNT) queda limpio (puesto a cero) tras la ejecución de la instrucción CLR (*clear*: limpiar).

En el capítulo anterior dijimos que el PC era un registro de 32 bits (aunque sólo 24 de estos bits son significativos). Esto significa que la dirección absoluta que especifica el operando puede constar de más de una palabra como en los dos ejemplos anteriores. ¿Cómo hace el ensamblador para saber cuánto espacio ha de reservar para la dirección absoluta? Sin duda sería un dispendio injustificado el tener una extensión de palabra larga por cada referencia de dirección absoluta, por ello lo que ocurre es que se emplea la extensión adecuada siempre que se conoce la dirección del operando (en el caso



de una referencia hacia atrás). En otras circunstancias habremos de especificar al ensamblador el empleo de las extensiones de palabra corta o palabra larga.

Volvamos al ejemplo de ADD para mostrar los efectos de una extensión de palabra larga:

Dir.:	Código:	Etiq.:	Instrucción:
1000	D679	ADD	HIDATA,D3
1002	0020		
1004	0000		

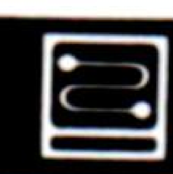
En este ejemplo la dirección absoluta de HIDATA es $200000. Nótese que el código para la parte ADD de la instrucción sigue siendo D6 pero que la parte de la dirección del operando ha cambiado de $78 hasta $79. En próximos capítulos veremos las formas para lograr esta extensión de palabra larga para el direccionamiento absoluto.

- *Direccionamiento por registro*: Es la manera más sencilla de direccionamiento en el 68000; en este caso el operando es uno de los registros del microprocesador. Por ejemplo, ADD D0,D3, donde la palabra D0 se añade al contenido de D3.

Hay algunas limitaciones en el empleo de este modo. Por ejemplo, no es posible tener un registro de dirección como destino de la instrucción ADD: así, no se acepta ADD D0,A4. Esto tiene arreglo si nos valemos de una instrucción diferente, ADDA D0,A4 donde ADDA es la instrucción de dirección de suma.

Si deseamos emplear palabras largas como objetos de datos en los ejemplos anteriores, deberemos incluir el atributo .L con la instrucción: ADD.L D0, D3 empleará las palabras de 32 bits enteras como objetos de datos.

- *Direccionamiento indirecto por registro*: Este modo es probablemente el más importante del 68000, dado que proporciona el *puntero* mencionado anteriormente y también los medios con los que se ejecutan las operaciones sobre la pila. Analicemos primero el empleo del puntero.



En la tarea de la programación necesitamos con frecuencia apuntar a un objeto de datos, ya sea un byte, una palabra larga o un objeto estructurado de datos como un registro o una tabla. Puede que, entonces, deseemos repetir el cálculo o la operación en otro miembro del mismo tipo de objeto de datos: es aquí donde el puntero resulta útil. El dibujo del puntero que adjuntamos muestra cómo puede ser empleado para dirigirse a diferentes elementos de un registro. Inicialmente el puntero dirige el cálculo que se ha de efectuar sobre el objeto de datos A; el puntero puede entonces ser restaurado para dirigir el cálculo que ha de efectuarse sobre cualquiera de los restantes objetos de datos.

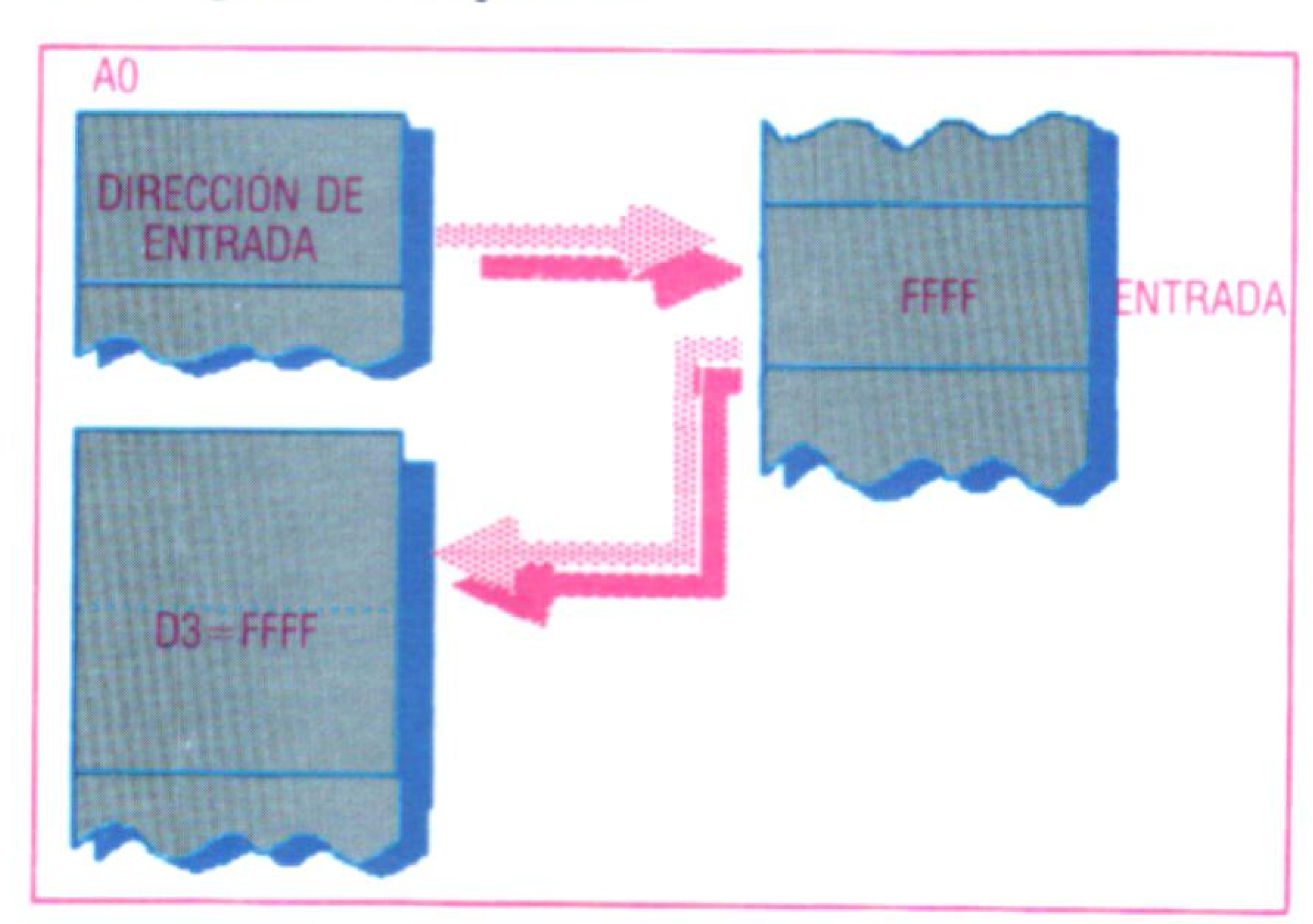
El elemento del 68000 a utilizar para lograr esto es un puntero de registro de direcciones, en vez del puntero almacenado junto a la instrucción, como en nuestro ejemplo general de modos de direccionamiento.

Esto puede comportar ligeros inconvenientes en algunas ocasiones, pero disponemos de ocho registros de direcciones.

Veamos ahora un ejemplo sencillo de modo de direccionamiento indirecto:

```
LEA      INPUT,A0
MOVE.W   (A0),D3
```

La instrucción LEA carga A0 con la dirección del objeto de datos de entrada llamado INPUT, el cual es seguidamente copiado en D3, como se muestra en el siguiente esquema:



Veamos ahora cómo se amplía este modo para operar sobre listas de datos. Ante todo, tenemos la *extensión de postincremento*. Se trata de que, una vez que se ha accedido al objeto de datos mediante el puntero, éste se incrementa para que apunte al contenido siguiente de la lista.

Examinemos el empleo de la extensión de posincremento en un ejemplo donde los objetos de datos son palabras:

```
MOVE.W   (A0)+,D3
```

Aquí el puntero en A0 es incrementado en dos unidades después que se ha accedido a la palabra apuntada por A0 y copiada en D3. Así, cuando apuntamos a la dirección $2000 originalmente, después de la operación MOVE.W veremos que A0 contiene $2002.

Es claro que si hemos empleado objetos de bytes entonces una operación MOVE.B sólo incrementará A0 en una unidad; y en cuatro con la operación MOVE.L.

La potencia de este modo de direccionamiento es obvia si se emplea en un bucle de programa para realizar algún cálculo en cada componente de una lista, por ejemplo. Así:



Cada vez que se realiza el cálculo se apunta automáticamente al componente siguiente de la lista tras cada ejecución de la instrucción MOVE.

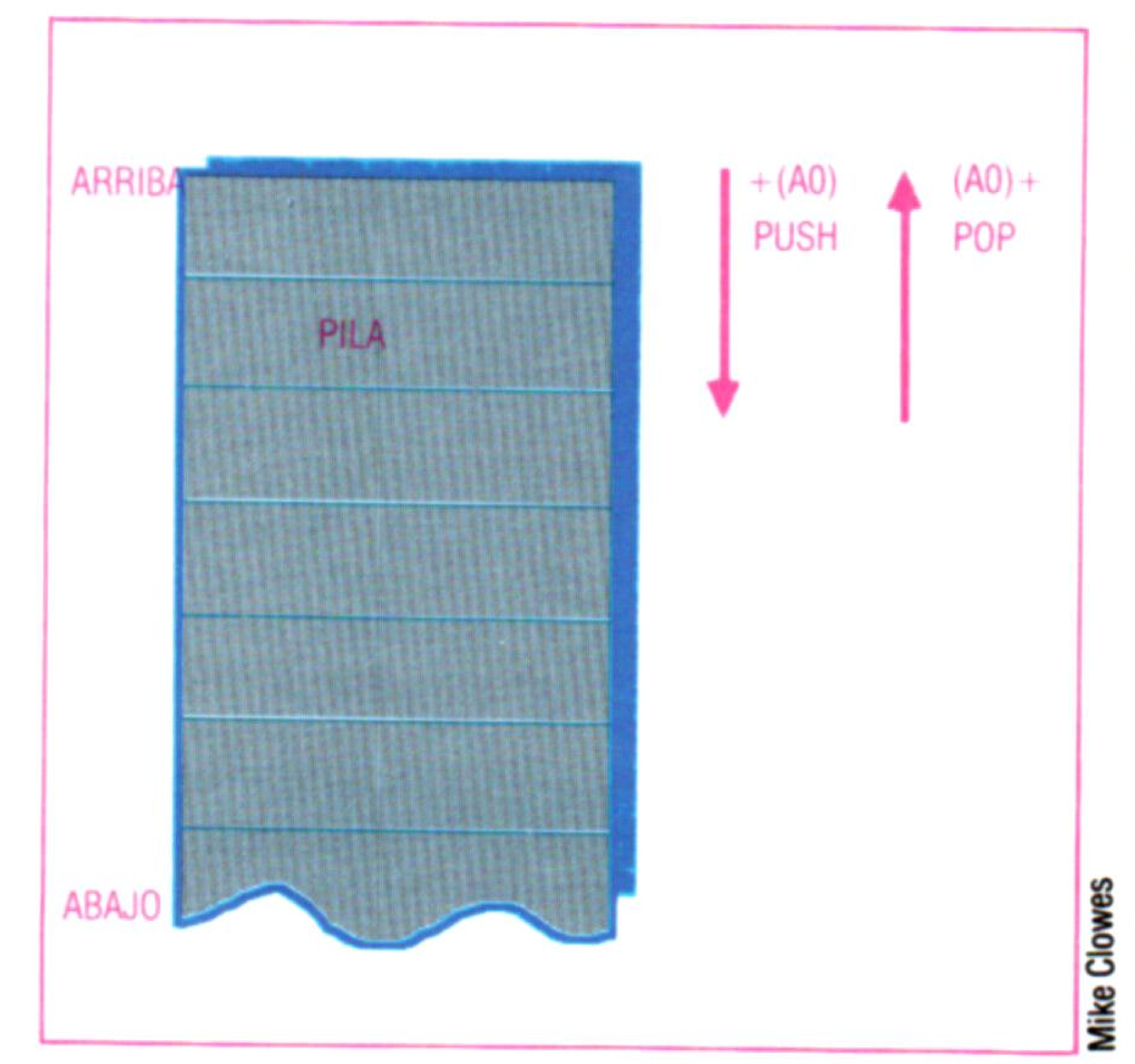
La otra extensión es denominada *predecremento*. En este caso el puntero de direcciones se decrementa antes de acceder al objeto de datos. P. ej.:

```
MOVE.W-(A0),D3
```

En este caso A0 se decrementará en dos unidades antes de ser empleado para copiar en D3 el objeto de datos.

El direccionamiento indirecto predecremental es el modo complementario del direccionamiento indirecto incremental. El posdecremento progresa a través de una lista a medida que se incrementan las direcciones, mientras que el predecremento funciona al revés. Sin embargo, no podemos separar las operaciones de *pre* y *post* según nos convenga. Por ejemplo no son admisibles ni (A0)− ni +(A0). Debemos, en cambio, respetar el predecremento −(A0) y el posdecremento (A0)+ especificados.

Habrá notado que en cierto sentido los modos de direccionamiento *pre* y *post* son operaciones de pila. Nos servimos de estas operaciones para "poner en" o "sacar de" una pila, y una pila es, como se define convencionalmente, una serie de direcciones de arriba hacia abajo. De esta manera, la instrucción MOVE D0,−(A0) "pone" (*push*), y MOVE (A0)+,D0 es una operación que "saca" (*pop*), tal como se aprecia en el siguiente dibujo.



Dirección apilada
Las instrucciones PUSH y POP, tan conocidas de los programadores del Z80, pueden simularse mediante las potentes extensiones predecremento y posdecremento del 68000, que pueden usarse para ejecutar un PUSH y un POP respectivamente

Más adelante estudiaremos las pilas y cómo se emplean en el 68000. De momento, baste con observar que disponemos de un modo de direccionamiento muy cómodo para acceder a listas de datos.

¿Cuál es el menú?

La inmensa gama de posibilidades que ofrece el "WordStar" lo convierte en un programa extraordinariamente eficaz

En el curso de la última década se han escrito literalmente centenares de programas para tratamiento de textos, y se ha dicho que cada uno de ellos representaba una mejora respecto a los demás. Pero el líder del mercado continúa siendo uno de los procesadores de textos más antiguos. El *WordStar*, de MicroPro, se escribió originalmente para máquinas CP/M e hizo su aparición en 1978, poco después de la creación de la compañía. Desde entonces se lo ha reescrito para operar bajo PC-DOS/MS-DOS, y fue elegido para el IBM PC y, por tanto, para sus compatibles, que ahora dominan el mercado de gestión. La fuerza que ha hecho resistir al *WordStar* son sus casi inigualadas facilidades para tratamiento de textos. Ello es así a pesar del hecho de que el sistema no es especialmente amable con el usuario, si bien fue pionero en el uso de menús de "ayuda" en pantalla.

Como acabamos de mencionar, las raíces del *WordStar* se hallan en el sistema operativo CP/M y, como tal, conlleva muchas de las mejores y peores características de ese sistema. En primer lugar, cuando se carga el programa se lo añade a la lista de programas "transitorios" del CP/M, lo que le permite sacar partido de las facilidades de operación de disco del CP/M. Y, al igual que el CP/M, el *WordStar* hace uso de una amplia gama de caracteres de control.

Tras cargar y ejecutar el programa, los usuarios del *WordStar* se encuentran con el Menú de apertura, que visualiza todas las opciones disponibles, así como el contenido de la unidad de disco que esté conectada en ese momento. Si usted no ha cambiado el disco, éste consistirá en los programas que componen el *WordStar*. Es interesante destacar aquí que los nombres de archivo del *WordStar* utilizan el mismo formato que los nombres de archivo CP/M: un nombre de ocho caracteres o menos seguido por un punto y una extensión de tres letras opcional.

En el Menú de apertura hay una lista de 13 instrucciones, que están divididas en cinco secciones. Bajo la denominación Instrucciones preliminares, usted tiene la opción de cambiar la unidad conectada, activar y desactivar el Directorio de archivos y establecer el Nivel de ayuda. El Nivel de ayuda por defecto tras el arranque es el nivel 3, y visualiza todas las instrucciones disponibles en la parte superior de la pantalla. Descendiendo hasta el nivel 0, la pantalla estará limpia a excepción de la línea de estado.

Capacidades esenciales

A continuación relacionamos 11 facilidades disponibles en muchos programas de tratamiento de textos. En este capítulo y en los siguientes veremos si los paquetes examinados disponen de estas facilidades y cómo están implementadas.

Desplazamiento de palabras
La capacidad de un programa para desplazar una palabra de una línea a la siguiente si no hay suficiente espacio.

Movimiento de bloques
Permite que el usuario defina un "bloque" de texto que se puede manipular independientemente del resto del documento.

Ayuda en pantalla
Ofrece asistencia, en forma de mensajes, informando al usuario sobre cómo acceder a las facilidades disponibles.

Pantalla de 80 columnas
Un paquete de tratamiento de textos debe otorgar al usuario la mayor visión posible de un documento. Para esto se considera que el mínimo es una pantalla de 80 columnas.

Contador de palabras
Indica al usuario cuánto ha escrito hasta entonces.

Buscar/reemplazar
Busca letras, palabras o frases y las cambia.

En esta etapa usted tiene la opción de abrir o editar un archivo de documento o uno normal. El primero es texto que se puede editar con el *WordStar*, mientras que el segundo comprende programas que se pueden ejecutar mediante su ordenador. A continuación sigue una lista de Instrucciones de archivo que incluye opciones para imprimir, cambiar el nombre, copiar y borrar.

Las dos últimas secciones del Menú de apertura son Instrucciones del sistema y Opciones WordStar. La primera le proporciona una interface con CP/M, permitiéndole ejecutar uno de los programas en disco o bien salir al sistema operativo. La segunda le ofrece la opción de usar ya sea *MailMerge* o bien *SpellStar*. El primero es un programa para correspondencia automática, ideal para imprimir cartas personalizadas, direcciones para sobres, y otras actividades de correspondencia propias de empresas. *SpellStar* es un verificador de ortografía que compara cada palabra entrada en un documento con un diccionario retenido en la memoria. Se destacarán todas las palabras que no posean correspondencia en el diccionario, para que el usuario las confirme.

En este punto, el empleo de la opción D hace que se abra un archivo de documento (o uno que se esté creando, si se trata de un archivo nuevo), y la pantalla volverá a pasar al Menú principal. En la línea de arriba aparece indicada la unidad conectada y el nombre del archivo de documento a editar. A ello le sigue la posición actual del cursor y un aviso que informa si la facilidad Insertar está activada o no.

Debajo de esto, suponiendo que el Nivel de ayuda aún esté establecido en 3, hay otra lista de instruc-

WYSIWYG
Éstas son las siglas de "*what you see is what you get*" (lo que ves es lo que tienes) y alude al proceso que permite contemplar el texto en la VDU en la forma en que aparecerá en la página impresa.

Facilidad para correspondencia
Instalados ya sea integralmente o bien como programas asociados, estos programas pueden "elaborar a medida" cartas o documentos estándares insertando nombres y direcciones específicos en los puntos requeridos e imprimiendo las etiquetas de las direcciones.

Verificador de ortografía
Proporcionados en formato similar al de los programas de correspondencia, los verificadores de ortografía leen cada palabra de un documento y la comparan con un diccionario retenido en la memoria. Se le señalan al usuario los vocablos que no concuerden con los del diccionario.

Tipos de letra disponibles
Aunque en gran parte depende de la impresora, muchos procesadores de texto soportan distintos tipos, tales como negrita y cursiva.

Unión de archivos
Los documentos están necesariamente limitados por la cantidad de memoria disponible. Por tanto, cuando se imprimen documentos extensos es útil poder unir los archivos para las funciones de impresión o Buscar/reemplazar.

Una galaxia de estrellas

Puesto que el *WordStar* se ha hecho tan popular, no es sorprendente descubrir que su fabricante, MicroPro, haya lanzado varias versiones del programa CP/M original. En los últimos años, una de las versiones más populares del *WordStar* ha sido *WordStar Professional*, que ha sido traducido para operar bajo el sistema MS-DOS utilizado en la gama Apricot y en el IBM PC. Como incentivo adicional, MicroPro ha incluido *SpellStar* y *MailMerge*. Más recientemente, la empresa ha lanzado el *WordStar 2000*, también para el IBM PC. Este programa, aunque en la superficie es similar a su antecesor, se ha descrito como "el Rolls Royce de los procesadores de textos". Proporcionado en cinco discos, los programas y archivos que constituyen el *WordStar 2000* añaden ¡hasta dos Mbytes! Pero esta potencia es cara. Por último, a modo de indicio sobre el futuro del *WordStar* CP/M, MicroPro ha escrito una versión del programa denominada *Pocket WordStar* destinada a usuarios del Amstrad CPC 464 y 664. Dado que estos ordenadores mantienen una visualización en pantalla de alta resolución, de los 64 Kbytes disponibles no queda memoria suficiente para poder ejecutar el *WordStar* adecuadamente. En consecuencia, MicroPro ha desarrollado para estas máquinas una versión más pequeña, que será comercializada por Cumana

ciones disponibles, nuevamente divididas en secciones. La primera visualiza un resumen de los movimientos del cursor, ilustrando ampliamente los lazos que unen al *WordStar* con el CP/M. Las primeras máquinas CP/M carecían de teclas para control del cursor y, por lo tanto, los movimientos se llevaban a cabo pulsando la tecla Control con otra tecla simultáneamente. Estas teclas están concentradas en el lado izquierdo del teclado, siendo su núcleo las teclas S, E, D y X, que corresponden respectivamente a cursor izquierda, arriba, derecha y abajo.

Entre los diversos elementos, usted observará B (CTRL B), que está listado como Reformar. Esto volverá a ajustar un párrafo en pantalla después de que la edición y las sucesivas correcciones hayan convertido al original en un tremendo lío. (Dicho sea de paso, el formato que aparece en la pantalla corresponde exactamente a cómo aparecerá en la copia, siempre y cuando, por supuesto, su impresora soporte los márgenes que se hayan establecido en el *WordStar*.)

A la derecha de la pantalla de ayuda Menú principal hay una serie de otros títulos de menú que le proporcionan acceso a otras funciones adicionales. El Menú de ayuda, por ejemplo, proporciona descripciones detalladas de cómo trabaja cada instrucción del *WordStar*. El Menú rápido, por el contrario, contiene diversas instrucciones "rápidas", incluyendo las numerosas opciones Hallar y reemplazar.

El Menú en pantalla proporciona los medios para formatear la pantalla a cualquier tamaño y forma, permitiendo establecer tabulaciones, márgenes, espaciado de líneas, etc. Una función muy útil permite establecer la longitud de la página, que aparece en el texto como una línea de puntos. Con ello puede verse dónde se producirán los cambios de página durante la impresión, y por lo tanto, puede distribuir consiguientemente el documento.

En el Menú en pantalla también está disponible el activador Wordwrap. Éste, una vez establecido, toma aquella palabra que no cabe al final de una línea y la coloca al comienzo de la siguiente, eliminando de este modo la molestia de tener que preocuparse por separar las palabras en sílabas mediante guiones.

El tercero de los menús adicionales contiene algunas de las instrucciones más útiles y potentes de que dispone el *WordStar*. Aunque el Menú de bloques puede llevar a cabo numerosas operaciones, su función primordial es tratar bloques de texto dentro de un documento. La gama de operaciones está listada bajo la sección Operaciones de bloques.

Mediante instrucciones CTRL al comienzo y al final del texto requerido se pueden crear bloques, que se visualizarán en "video invertido" (el texto del bloque aparecerá con un brillo considerablemente menor que el resto del texto, para su fácil reconocimiento). Una vez creado el bloque, se puede trasladar o copiar el texto en otra zona del documento, suprimirlo o incluso guardarlo en un archivo separado en disco. Pero usted no se ve limitado a desplazar párrafos solamente. Con el *WordStar* también se pueden desplazar columnas, lo que resulta muy cómodo cuando se está trabajando con tablas o cartas con dos o más columnas de texto. En este caso, todo es cuestión de demarcar columnas de bloques en lugar de líneas.

En el Menú de bloques se incluyen tres instruccio-

Desplazamiento de palabras

Si bien el *WordStar* desplaza las palabras cuando se está escribiendo el texto, cuando se cambian los márgenes no vuelve a formatear el texto de forma automática.

Movimiento de bloques

El *WordStar* permite la definición de bloques de cualquier tamaño y la totalidad de la gama de manipulación de bloques.

Ayuda en pantalla

Los menús de "Ayuda en pantalla" son característicos del *WordStar* y contribuyeron a acrecentar su popularidad.

Pantalla de 80 columnas



El programa no sólo soporta una pantalla de 80 columnas, sino que también permite márgenes de hasta 255 caracteres.

Contador de palabras

El *WordStar* no posee facilidad para contar las palabras.

Buscar/reemplazar



El *WordStar* soporta varias versiones de esta facilidad, permitiendo la búsqueda de series de hasta 30 caracteres.

WYSIWYG



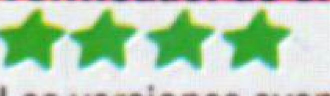
Más que ningún otro paquete para tratamiento de textos, el *WordStar* es famoso por su capacidad para formatear el texto en pantalla.

Facilidad correspondencia



El *WordStar* fue uno de los primeros que incluyó un programa para correspondencia, y el *MailMerge* sigue siendo el estándar que sirve de referencia a los demás.

Verificador de ortografía



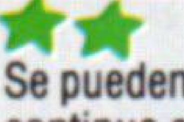
Las versiones avanzadas del *SpellStar* poseen un diccionario de alrededor de 20 000 palabras para la verificación ortográfica.

Tipos de letra disponibles



Los tipos de letra adicionales incluyen negrita y cursiva.

Union de archivos



Se pueden imprimir de modo continuo archivos separados utilizando el *MailMerge*, pero no se soportan facilidades tales como "Buscar/reemplazar" a través de varios archivos al mismo tiempo.

À la carte

El *WordStar* se caracteriza por los menús de ayuda, que se visualizan en la parte superior de la pantalla. Aunque los usuarios ya experimentados tienden a prescindir de ellos, son sumamente útiles para los principiantes, porque proporcionan una fácil referencia a las instrucciones disponibles.

Cada una de las tres pantallas que vemos aquí proporciona una gama de instrucciones diferente. La pantalla de apertura, que es la que se presenta al usuario cuando se carga el *WordStar*, contiene mayormente las instrucciones DOS que le permiten acceder a los archivos. El menú principal contiene instrucciones que con toda probabilidad se usarán cuando se digite el texto, mientras que el menú de bloques contiene varias instrucciones de edición







Le dice al usuario cuál es la unidad conectada y cuál el archivo que se está editando

Éste es el menú principal bajo el nivel de ayuda tres, dando explicaciones sobre las instrucciones disponibles

Números de página, línea y columna indican la posición del cursor

Este aviso recuerda que está activada la mod. INSERTAR

Las versiones IBM del *WordStar* permiten entrar numerosas instrucciones a través de las teclas de función. Esta línea muestra la configuración actual de las teclas de función

Esta línea muestra la cantidad de columnas establecidas, si bien la misma se puede modificar. El signo ! indica los ajustes TAB por defecto, que también se pueden alterar


nes para guardar, ofreciendo las opciones de retornar al texto, ir al menú principal o salir del *WordStar*. Usted debe tener presente que cuando se utilizan instrucciones incluidas en los menús separados no es necesario volver a llamar el menú a la pantalla. Para guardar y salir del sistema, por ejemplo, sólo debe digitar CTRL KX, que ejecutará la instrucción automáticamente.

Vamos a concluir esta breve reseña del *WordStar* con una análisis de las *instrucciones de punto*. Éstas están incorporadas en el texto en líneas separadas, empezando cada una de ellas, como parece natural, con un punto, y están relacionadas con el formateo de la impresión y la adición de características a la salida impresa, como anchuras de los márgenes superior e inferior y números de página. La instrucción .pl (seguida por un número), por ejemplo, establece el número de líneas a imprimir en una página. Debido a que el *WordStar* no actúa sobre estas instrucciones hasta que el software esté procesando el documento para su impresión, en realidad actúan como caracteres de control para la impresora.

La potencia del *WordStar* ha conseguido mantener este programa a la cabeza de su especialidad durante más de una década. Dado que el CP/M se "extiende hacia abajo" del mercado, hacia máquinas personales tales como la gama Amstrad, parece probable que el *WordStar* se convierta también en el líder del mercado de ordenadores personales.

Un paquete popular

El *WordStar* constituye un interesante ejemplo de cómo la reconocida popularidad de un programa genera su propio impulso para hacerse aún más popular. Una vez que el *WordStar* se estableció como líder del software para tratamiento de textos para micros CP/M, muchos fabricantes comenzaron a empaquetarlo en sus máquinas, lo que condujo a una base de usuarios aún mayor para el paquete. Un temprano ejemplo de este tipo de comercialización fue el Osborne 1, máquina que no sólo tenía empaquetado el *WordStar*, sino también la hoja electrónica *SuperCalc*. El hecho de que ambos programas operaran bajo CP/M significaba que podían compartir archivos comunes. Esta idea demostró ser tan popular que otros programadores comenzaron a incorporar el concepto de pasar y compartir archivos, lo que finalmente condujo al desarrollo de paquetes de software integrado, como el *Lotus 1-2-3*. En la actualidad, casi todos los micros vienen con software empaquetado, en especial, procesadores de textos. Los mismos van desde el sofisticado *Perfect Software* de Thorn EMI (para el Advance y el Wren) hasta el económico Tasword II, que se empaquetó en el *six-pack* que se entregó junto con las primeras versiones del Spectrum+


NUEVO
Para los jóvenes de 9 a 90 años
SHERLOCK HOLMES
Una nueva y divertida serie de tve realizada en COMICS forum
PÍDELA EN TU QUIOSCO
COPYRIGHT © 1986 RAI/PAGOT/TMS
A-048 B


GRAN NOVEDAD EDITORIAL
PARA TI, PROFESIONAL EN ACTIVO.....
PARA TI, FUTURA SECRETARIA.....
enciclopedia de la
SECRETARIA
de
PLANETA-AGOSTINI
PIDELA EN TU QUIOSCO
o suscríbete ahora
llamando al (91) 415 97 12
... para estar al día
... para ser más eficaz
... para actualizar tus estudios
... para encontrar mejor empleo